Technical Report 1229

MARVEL: A System for Recognizing World Locations with Stereo Vision

David J. Braunegg

MIT Artificial Intelligence Laboratory

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
AI-TR 1229

2. GOVT ACCESSION NO

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
MARVEL: A System for Recognizing World Locations with Stereo Vision

5. TYPE OF REPORT & PERIOD COVERED
technical report

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
David J. Braunegg

8. CONTRACT OR GRANT NUMBER(s)
N00014-85-K-0124
DACA76-85-C-0010
N00014-86-K-0685

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

10. PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

12. REPORT DATE
May 1990

13. NUMBER OF PAGES
245

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Office of Naval Research
Information Systems
Arlington, VA 22217

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

recognition
mobile robot
location recognition
stereo vision
navigation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

To use a world model, a mobile robot must be able to determine its own position in the world. To support truly autonomous navigation, I present MARVEL, a system that builds and maintains its own models of world locations and uses these models to recognize its world position from stereo vision input. MARVEL is designed to be robust with respect to input errors and to respond to a gradually changing world

(continued on back)

Block 20 continued:

by updating its world location models. I present results from real-world tests of the system that demonstrate its reliability. MARVEL fits into a world modeling system under development.

MARVEL: A System for Recognizing World Locations with Stereo Vision

by

David Jerome Braunegg

B.S. Elec. Eng., University of Pittsburgh (1980)
M.E. Elec. Eng., University of Tennessee (1983)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

June 1990

MARVEL: A System for Recognizing World Locations with Stereo Vision

by
David Jerome Braunegg

Submitted to the Department of Electrical Engineering and Computer Science
on May 2, 1990, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

To use a world model, a mobile robot must be able to determine its own position in the world. To support truly autonomous navigation, I present MARVEL, a system that builds and maintains its own models of world locations and uses these models to recognize its world position from stereo vision input. MARVEL is designed to be robust with respect to input errors and to respond to a gradually changing world by updating its world location models. I present results from real-world tests of the system that demonstrate its reliability. MARVEL fits into a world modeling system under development.

Thesis Supervisor: W. Eric L. Grimson
Title: Associate Professor of Computer Science and Engineering

Acknowledgments

First and foremost, I want to thank Patty Cox for all the love, support, and encouragement that she has given me during the writing of this thesis. She helped with the typing and proofreading, which is usually a thankless job. Thank you. (Any remaining errors in this thesis were put in by me after she finished.) Most of all, however, she taught me the importance of balancing work and play and helped me to find that balance. While this thesis work has taught me many things, I will carry the perspectives that she helped me learn for the rest of my life.

My parents, Hubert and Susan Braunegg, have been supportive over my many years of formal education. Although they did not understand at times why I spent so much time in school instead of the real world, their love never failed.

My sister Jan, through our long phone conversations, was a reminder of what life was like outside of school. Always there when I needed her, she helped me to keep a perspective on the big picture of life when I was buried in my own little world.

Jeannine (Haas) Bell, Lisa Demichele, Lisa Fucile, Mike Bell, Anita Moskowitz, and Drew Shattuck were all a second family to me. Their friendship has brought much joy to my life.

Pam Melroy has been an endless source of love and support for many years. Although necessarily at the other end of a telephone wire, she helped me through the bad times and celebrated the good times with me.

Scott Gorsuch listened when I needed to talk, which helped a lot.

My advisor Eric Grimson deserves thanks beyond my capacity for expressing it. He showed unbelievable patience over the course of my graduate career and I was constantly amazed that he still had more to give. Without that patience I never would have earned my Ph.D.

The group meetings with Eric Grimson and my fellow graduate students under him helped increase my understanding of computer vision and recognition. Tao Alter, Todd Cass, David Clemens, David Jacobs, and Sandy Wells also read drafts of various thesis chapters. Paul Viola suggested the hardware speedups discussed Chapter 12 and also helped in reading thesis drafts.

My committee, consisting of Ellen Hildreth and Rod Brooks as well as Eric Grimson, helped me shape this research and thesis into something of which I can be proud.

Patrick Winston provided a wonderful atmosphere in the AI Lab that encouraged creativity.

I am sure there are people I have forgotten and for this I apologize. The hectic times of finishing a thesis can cause one to lose one's memory, if not one's mind.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124 and under Army contract number DACA76-85-C-0010 and in part by the Office of Naval Research University Research Initiative Program under Office of Naval Research contract N00014-86-K-0685.

To Patty

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pcople have always wanted someone *else* to do the work. First there were slaves, then indentured servants, and finally immigrant labor. Now that robots are seeing industrial use, people are asking if a mechanical helper might be feasible. Even Newsweek has expressed a desire for a "home helper" robot:

> And what of the long-promised home robot? This little electronic servant, capable of delivering a frosty beer from the fridge, picking up the kids' toys and washing the occasional window, ...
>
> Michael Rogers
> *Newsweek*
> December 25, 1989

It is indeed true that we can build mobile robots that would perform many household tasks. That same Newsweek writer, however, also put his finger on an important problem:

> ... probably won't be a mass-market item in the '90s—unless we modify our homes to accommodate them. Every room would need to have tiny radio beacons to tell the robot where it is.

We cannot make a fully autonomous mobile robot. Yet. One requirement for autonomy is that a robot be able to determine where it is. This information is obviously important for a mobile robot since it could be anywhere in its environment and the tasks it must perform depend on just where in that environment it is. Because we do not wish to modify our homes (for example, by installing beacons) to accommodate this mechanical servant, however, it should be able to determine its location in places *as they exist now*.

To achieve true autonomy, a mobile robot must be able to determine its own location. In the same way that an autonomous robot should not need us to modify the environment to accommodate it, the robot also should not require us to lead it by the hand and show it its environment. An autonomous mobile robot should be able to *explore* its environment, remember what it has discovered, and use this information. In the context of determining where it is, a mobile robot should be able to go to new places, remember the places it has been, and tell when it is in one of these places again [Braunegg 1989c].

This thesis describes a system designed and built to support autonomous mobile robot navigation in exactly the way just outlined. MARVEL (Model building And Recognition using Vision to Explore Locations) is a system that uses stereo vision input to sense its environment, builds models of locations it has encountered, uses these models to recognize when it is once again in these locations, and updates these models over time. This work recognizes that any sensing system will make mistakes. Since MARVEL uses its sensor information to build its own models, the data and models used for recognition may contain errors. MARVEL deals with these errors both through a robust recognition system and by updating the information contained in its models. The model update process also enables MARVEL to deal with an important aspect of the real world: *change*. As a location in the environment changes, the model used to represent that location will become less and less correct unless the model also changes. MARVEL explicitly reacts to a changing world by updating its models based on new sensor data. Finally, MARVEL is designed to operate on a *mobile* robot. The strategies it uses for recognition take advantage of that mobility.

1.1 Problem Description

This thesis gives one answer to an important question in robot navigation: How can an autonomous mobile robot determine where it is, based on data it has collected and models it has built, and do this in spite of a changing world and sensor error? The answer presented is MARVEL and this chapter gives the background information necessary to understand this solution.

This work is concerned with recognizing locations in the world. The words *world*, *location*, and *position* will be used throughout this thesis with specific meanings. *World* will be taken to be the environment in which the mobile robot resides, consisting of all the places the robot might be. *Location* will be taken to mean the place in which the mobile robot currently resides, such as a room. This use of the word location follows from the definition [Mish 1986]:

> **location**: a site occupied ... or marked by some distinguishing feature.

Position will be taken to mean the exact spot where the robot can be found in a location, including the robot's orientation (i.e., rotation) with respect to some fixed reference point in that location [Mish 1986]:

> **position**: the point occupied by a physical object.

1.1.1 The Problem Domain

As stated earlier, this thesis is concerned with supporting navigation for a mobile robot by recognizing locations. Specifically, I address the question of how to support navigation of an autonomous mobile robot over a long period of time with no human intervention.

Autonomous Mobile Robot with Long Lifetime

Simply stated, a mobile robot is a self-propelled movable platform. The platform supports sensors and manipulators or other means to examine and interact with the world. Mobility allows these sensors and manipulators to be transported to the place they are required and then relocated as need be. A typical mobile robot [Brooks 1986] has motor-driven wheels for driving over smooth terrain, the abilitv to steer, and often the ability to turn in place. Mobile robots come in a host of shapes, sizes, and configurations (e.g., [Nilsson 1969], [Giralt *et al.* 1983], [Moravec 1983], [Brooks 1986]).

Autonomy simply means that the robot should function without human intervention. This simple statement has important consequences, however, if we wish to attain true autonomy. First, "no human intervention" implies that no one will lead the robot around to "teach" it the world. Thus, the robot must "learn" the world. In the context of this thesis, the robot must learn how the locations it encounters look so that it can recognize them when they are encountered once again. This learned representation is a *model* for that location. Second, no human will tell the robot if these models are correct, so it must update the models on its own, based on the current sensor data. The correctness of a model is related to how well it represents a given location and how free it is from error. Essentially, then, autonomy can be thought of as self-sufficiency for the robot.

Long lifetime for a mobile robot means that it will exist for days, weeks, or months at a time in its world. This contrasts with *mission-level* autonomy, where the robot must only go out, perform a specific task, and (usually) return. The long lifetime implies that the system must be robust, since there will be many opportunities for it to fail. Thus, the models built must be generally correct. Also, they must allow changes, since the locations they represent may change over time. Finally, recognition must be reliable, since the ability of the robot to fulfill its continuing mission will depend on its knowing where it is. A long lifetime demands robust systems and provides an acid test for those systems: over time, if something can fail, it will. Robustness will be achieved through good engineering and design and through providing methods for recovery from error.

Indoor, Office-Type Environment

Mobile robots can be designed for many different environments, each with its own set of requirements and constraints. Some general constraint can be placed on the environment, however, to make the location recognition task possible.

If one looks at the world and then leans over and looks again, the world appears different. This everyday occurrence implies that MARVEL should have a level floor or a tilt sensor on board the robot so that views taken of the world at different locations can be compared based on the same concept of what is vertical. Also, we do not require that the world remain static. A gradually changing world where most of what is seen today persists tomorrow is allowed and is explicitly considered by the model update process (Chapter 7).

These simple requirements are met in office-type environments. The floors are level and the large structures in rooms do not change often or by much. In order to test the ideas about recognition and model building being explored in this thesis, then, the robot's world has been limited to indoor, office-type environments. Testing of the robot, which is described in Chapter 8, was performed in the ninth floor area of the MIT Artificial Intelligence Laboratory. (Extensions to other environments will be discussed in Chapter 12.)

Stereo Vision

As stated earlier, to explore the investigation of location recognition and model building and maintenance to support recognition, a stereo vision system has been chosen to supply the input data to MARVEL. This input system is explained more fully in Chapter 2, but a few of the assumptions and consequences of choosing stereo vision for acquiring input data will be detailed here.

Stereo vision algorithms operate by comparing two different views of the world taken from different positions. By knowing the displacements between the two viewing positions, the images can be compared and three-dimensional information obtained for parts of the world visible in both images. Specifically, this information is returned as robot-centered three-dimensional x-, y-, and z-coordinates of visible parts of the world.

The specific stereo algorithm used is the Marr-Poggio-Grimson stereo algorithm [Marr and Poggio 1979] [Grimson 1981] [Grimson 1985], with enhancements [Braunegg 1989a] [Braunegg 1990] (see Chapter 2). This stereo algorithm is based on *features* found in the left and right images. The specific features used are intensity edges. Thus, the algorithm yields a sparse set of three-dimensional data at places in the world that correspond to these image features.

The information obtained from stereo vision typically has good angular resolution from the viewpoint. The distance resolution, however, decreases with distance of the matched features from the viewpoint. These resolution uncertainties are well-understood [Matthies and Shafer 1986] and are taken into account in the representation used for the input data (see Chapter 4). *Any* sensing of the real world, however, will have errors. There are both errors of commission and errors of omission. Errors of commission (or hallucinations) occur when the input system returns data that does not correspond to anything in the world. Errors of omission occur when something is present in the world but no data is returned for it. Any system that uses real-world data, then, must explicitly accept the fact that these errors will occur and must deal with them. MARVEL handles these errors through the model update process (Chapter 7).

In contrast to the standard recognition problem, wherein an outside observer attempts to recognize an object, a mobile robot is *inside* of the object (the location) that it is trying to recognize. In order to obtain sufficient information about the location, we assume that the stereo cameras are mounted on a rotatable platform. Either the robot itself must be able to rotate in place, or the cameras must be mounted on a turntable on the robot. The consequences of the rotation method used are discussed in Section 2.3.5.

1.1.2 A Motivational Example

Given all that has been written above about the problem being addressed in this thesis, what would be a practical example of this problem in the real world? Some day in the future you will be able to walk into Sears and buy *Cleano*—the Kenmore House Cleaning Mobile Robot. When you get Cleano home, you will unpack it and turn it on. Cleano's job will be to keep your house clean by vacuuming the rugs and picking up objects left lying on the floors. Being a fully autonomous robot, Cleano learns the layout of your house without requiring you to give it a Cook's tour. Since it cannot know the specifics of your house, however, you must tell it certain things. For example, I would take it to the study and tell it not to clean this room (because my thesis-in-progress is spread all over the floor). Once given this information, Cleano, being able to recognize the study again after having seen it once, will avoid doing any work in the study. Taking Cleano to the playroom, I show it the corner of the room where the toys are kept. Cleano then remembers this specific place in this specific room as the place to return any objects found on the floors. After receiving these instructions, which are specific to its new home, Cleano goes along merrily about its business of vacuuming rugs and picking up stray objects without my ever having to worry about it again. It builds its own map of my house and uses this map to keep track of which rooms have been cleaned and how long ago this occurred. It uses its recognition capability to keep track of its location and basically never needs my help again.

1.1.3 Challenges to Such a System

There are certain challenges to any system designed to support navigation by a mobile robot through recognizing locations. Input errors will occur—MARVEL is robust with respect to them. The world will change—MARVEL changes its location models to respond to these changes. New locations will be encountered—MARVEL builds its own models to be able to recognize them. Sensing deficiencies will occur—MARVEL uses various sensing strategies to overcome them. All of these issues and more will be discussed in the following chapters.

1.2 Related Work

Most work to date that has addressed mobile robot travel has either been concerned with planning collision-free paths or modeling the world to allow planning of larger-scale routes through the environment. Little has actually been done to enable a mobile robot to recognize its current location.

The closest work to MARVEL was that of [Drumheller 1987], which addressed the localization problem rather than recognition. Using an office-type environment, Drumheller assumed that the current location was already known. The given task, then, was to determine the position and orientation of the robot within this location as accurately as possible. The models used were previously constructed offline and supplied to the system. 100 range readings from a sonar rangefinder were obtained in 3.6° incremental rotations about the current position. These readings were compared to the model using the method of [Lozano-Pérez and Grimson 1984]. In testing on real data, 17 of 24 localizations were correct and yielded the robot position within ±1 foot and ±5°. No attempt was made to extend the method to recognition against a database of models.

There are a few works that specifically addressed the problem of recognizing world locations, but none fulfilled the goals described earlier. [Kuipers and Byun 1987, 1988] described a simulation of a robot exploration and map-learning system that has a sonar-based recognition subsystem, but not many details were given. This system has not yet been extended to real data. [Kadanoff 1987] used infrared coded beacons to mark world locations. This method is unacceptable where one does not want to modify the world to accommodate a robot. [Nasr *et al.* 1987] used high-level landmarks such as telephone poles to check a robot's progress as it followed a road. Such easily-identifiable landmarks are rare in office-type or natural outdoor environments. [Faugeras *et al.* 1986] used common stereo features in successive views as their robot moved to correspond the views while map making, but they did not do independent recognition based on these features.

Some related work is being done on the Hilare mobile robot project at the *Labo-*

ratoire d'Automatique et d'Analyse des Systemes (LAAS) in Toulouse, France. [Laumond 1984, 1986] defined rules for determining rooms and doorways, but recognition of a location was solely based on determining the robot's position by following a previously-constructed world model. The world model used was based on data from a laser range finder and from optical shaft encoders connected to the robot's wheels [Chatila and Laumond 1985]. No method, however, was given to deal with the cumulative error that would build up in this data. [de Saint Vincent 1986, 1987] used vertical lines from a stereo vision system for tracking and correcting a robot's position with respect to a map. The proposed recognition system assumed that known features were visible to constrain the matching problem; also, no recognition algorithm was given.

Early work in modeling the world centered on cognitive maps, i.e., the representations that humans use to model the world. [Piaget and Inhelder 1967] performed early studies on how children represent the world. [Lynch 1960] in his work on city forms investigated the cognitive maps that people create and use for urban environments. [Kuipers 1977] developed a computational model of cognitive maps used for large-scale environments. This work was motivated in a large part by the work of Lynch. Unfortunately, Kuipers' work has remained in the realm of computer simulations and has not been applied to real-world domains (e.g., [Kuipers and Byun 1987, 1988]). In general, cognitive maps as noted above tend to require a great deal of information from their underlying perception systems and also a comprehensive reasoning facility to interpret the perceptions. These lower-level systems have not yet reached the level of sophistication needed to make cognitive maps feasible.

Other schools of thought have approached the problem of world modeling from the viewpoint of robotics without concern for the representations humans might use. For example, occupancy grid methods represent the entire world via a two- or three-dimensional grid, with object locations marked on the grid. The early work of [Moravec 1983] on the Stanford Cart and the CMU Rover represented the three-dimensional locations of feature points obtained from a stereo algorithm on such a grid. Objects were simply represented as clouds of feature points.

The grid methods can be extended by labeling each cell of the grid as either occupied or unoccupied. These labelings can also be associated with a confidence value. [Moravec and Elfes 1985] have used such a grid method for mapping the world based on range data from ultrasonic rangefinders. Their implementation used a two-dimensional grid with 6-inch square grid cells. Cells were marked as either occupied or empty, each with a confidence between zero and one. Information from sensor readings at various locations was merged with the existing data in the grid map, based on these confidences. This work was extended by [Serey and Matthies 1986] to build the world map from a one scanline stereo algorithm.

The preceding grid-based models were two-dimensional. This restriction is common for ground-based mobile robots whose motions are constrained to take place in the plane of the floor. Grid methods were extended to three dimensions by [Stewart 1988], however, to model the world inhabited by an underwater robot whose movement has six degrees of freedom. Each cubic volume element, or voxel, of the grid had an associated feature vector that represented the sensor data within that region. Video and sonar data were represented, although other sensor information, such as chemical properties and temperature of the seawater, could also be included. The feature vectors that described this data used stochastic techniques to model the uncertainty in the measurements and to merge new information with the existing data.

Grid methods have been used successfully for planning robot trajectories in small worlds or in small pieces of large worlds, but do not provide the abstract information needed to plan routes through different locations in a large world. These methods also depend on maintaining an exact metrical map of the world in the presence of sensor error and uncertainty. There is an implicit dependence on the ability to perceive all of the known world from any location in it in order to maintain these maps.

Configuration space, frequently used to represent obstacles for manipulation planning [Lozano-Pérez 1983], was used by [Brooks and Lozano-Pérez 1985] to find collision-free paths for a polygonal moving object. In their algorithm, the moving object (the mobile robot) moved in the plane and thus had one rotational and two translational degrees of freedom. Obstacles were considered as polygonal objects. This work assumed *a priori* knowledge of the locations of the obstacles in the environment as well as the start and goal locations. In this thesis, however, I am concerned with moving through environments that have not been previously encountered. Also, configuration space does not provide a high enough level of abstraction to plan routes through large-scale space that traverse many locations.

Freespace was represented by generalized cones in work done by [Brooks 1983]. The volume swept by the mobile robot as it translated and rotated was characterized as a function of its orientation. The swept volume of the robot was then compared with sweepable volumes of freespace in the environment to find paths (freeways). This representation was extended in [Kuan *et al.* 1985] by representing large open spaces, which are not naturally described by generalized cylinders, by convex polygons.

The freeway representation was combined with explicit symbolic reasoning about uncertain quantities in [Brooks 1984]. In this work, the range of locations that a robot may occupy after an uncertain movement was represented by an uncertainty manifold. The uncertainty manifold for the current location of the robot grew due to cumulative error as the robot moved until there was reason to believe that the robot had returned to a previously encountered location along its path. The location previously stored was used to restrict the current uncertainty manifold and this restricted uncertainty was back-propagated along the robot's path through the uncertainty manifold stored for previous locations. The mathematics for the uncertainty manifolds was further developed in [Brooks 1985] and a cylinder approximation in configuration space was given for the manifolds. As with the mapping methods previously mentioned, freespace/freeways do not allow us to abstract to the notion of a location, thus requiring us to plan all of our movements explicitly before moving at all.

[Chatila and Laumond 1985] used three kinds of models to represent the world: a geometric model, a topological model, and a semantic model. The geometric model contained data directly received from the sensors. This model covered the known world and the data was referenced to an absolute coordinate frame. The information in this model was updated and maintained "as accurate[ly] as possible." The topological model represented the topology of the known world and consisted of places (represented by convex polygons constructed from the geometric model) and connectors (common edges between places). The semantic model consisted of meanings (labels) for places and was distributed between the geometric and topological models. As with any method that attempts to maintain geometric data over the entire known world, cumulative errors would become a problem as the size of the modeled world grows. Paths planned between two distant places based on this stored geometrical data would contain sufficient error to make them difficult to use if not useless.

[Faugeras *et al.* 1986] addressed the problem of building maps for navigation from stereo measurements. They described the world in terms of point, line, and plane primitives obtained from the stereo input. For each position of the robot, they built a three-dimensional description of the environment that contained the positions and orientations of these primitives along with uncertainty parameters describing them. An Extended Kalman Filter was used to relate two different frames of reference via primitives that the two world representations had in common. The Kalman Filter enabled them to estimate the transformation between the frames of reference and to estimate the uncertainty of this transformation. The transformation was then used to update the descriptions (geometry and uncertainty) of the primitives common to the world models associated with the two reference frames. This last step of "closing the loop" by updating the descriptions of the primitives was intended to reduce the cumulative error associated with the model as it was built over time. The errors considered, however, were only those of position. Faugeras, Ayache, and Faverjon ignored the extraneous data that arise from the output of a stereo vision system. Both these extraneous data and objects once present but later missing in the world would have to be removed from the representation, yet this question was not addressed. For the Kalman Filtering it was also assumed that there was some estimate of the displacement between the locations of the robot for successive data sets. This implies that the model for a given location must be complete when it is first built. If the same location were encountered at some future time, no method was given to estimate the robot's position within that location (with respect to the stored model) and so new data gathered at this future time could not be incorporated into the stored model.

[Kriegman *et al.* 1989] described a system using stereo vision to map indoor hallways. The purpose of this system was to provide a representation of a hallway that would allow a mobile robot to navigate that hallway. This system developed a 2-D representation by using stereo vision based only along the horizon lines in the left and right images. Intensity edges found in the horizon lines were matched and 3-D coordinates were found for them. The system was only interested in labeling the locations of doorways along the hallway walls. Because of this, markings, fuseboxes, etc., whose edges met the criteria for a door in the one line of stereo were labeled as doorways. These criteria included restricting assumptions such as contrasting colors to separate doors from walls and high-contrast door edges. Both these criteria are not met by the doors in the MIT AI Laboratory, for example, because both the doors and walls are white and the aluminum doorframes do not yield high-contrast edges. Because the system only considered one line (the horizon line) of the image, edge length could not be used to distinguish door edges from other edges. This same restriction also prevented the system from detecting any obstacles not at the height of the cameras. Finally, the modeling system assumed that there would be no extraneous data and provided no method for removing any such data that the model might acquire.

[Sarachik 1989] described a method of building maps of office-type environments by finding rooms and the doorways connecting them. The rooms had to be rectangular and were detected by finding the four wall/ceiling junctions. Doorways were identified by finding two (door) edges with a more distant intervening edge. A world map was proposed that consisted of a series of rooms connected by doorways. A room's identity would be determined by its dimensions and the paths that connect it to the rest of the world model. The "roomfinder" was implemented, but seldom ranged all four walls correctly during one sampling of the room. The "doorfinder" was also implemented, but only half of the doorways it found were true doorways. The strength of this work was the ability of the system to determine that it was inside a rectangular room, which it did well for rooms with sufficient wall/ceiling contrast.

A method for learning a world model was presented by [Mataric 1990]. A ring of sonars surrounding a mobile robot determined the distances to the walls of a room and an onboard compass gave bearings used to help identify the walls. There were four types of landmarks: left wall, right wall, corridor, and "junk" (the absence of a landmark). The world was modeled by a graph, each node of which consisted of a landmark, its average compass bearing, a rough length estimate, and a global position estimate that was derived by integrating the compass measurements. The robot built a room model by following the walls of a room and creating a new node for each new landmark (i.e., wall) encountered. Localization with respect to this model was only possible up to the determination of which landmark (wall) was nearby. To determine that it had returned to its starting point after circumnavigating a room, the robot depended on its global position estimate as well as on the landmark type (e.g., left wall facing north) of the starting point. No description was given of how this world model could be extended to represent more than one room or how several room models could be combined. Also, doors that can be either opened or closed could present a problem during wall following. Since a piece of furniture can be mistaken for a wall, this method may not be robust in dealing with changes in a room's furniture arrangement. Finally, the method does not provide a way of modeling the open space in the center of a room.

Other representations of the world have also been used to support path planning. Visibility graphs store all possible path segments between the vertices of obstacles (e.g., [Lozano-Pérez and Wesley 1979]). Planning is accomplished considering only these paths. Variants of these continuous paths between object vertices include paths with continuous tangents and paths with continuous curvature. Voronoi diagrams have been used for "safest" path planning (e.g., [Brooks 1983]). The Voronoi diagram consists of all points that are equidistant between adjacent obstacles. Planning is performed along the set of paths thus formed by these points, yielding paths with the maximum clearance between obstacles. Potential fields have also been used for path planning (e.g., [Arkin 1987]). Obstacles are represented as hills in the potential field and the goal position is represented as a valley. Planning is then accomplished by tracing a path through the potential field so as to minimize the "energy" required to move from the start position to the bottom of the goal valley. Visibility graphs, Voronoi diagrams, and potential fields all assume *a priori* knowledge of the world to be modeled and are thus not well-suited to planning paths through a world that is being explored. Also, since these methods depend on having the locations of all objects in the world, cumulative errors prevent their use when dealing with large worlds.

1.3 Goal of the Thesis

This thesis addresses the problem of recognizing world locations to support mobile robot navigation and how this can be accomplished autonomously. The solution presented to this problem is a complete, working system named MARVEL. More than describing a system, however, this thesis addresses issues involved with the location recognition problem. Such issues can best be understood through the insights gained via the design and construction of a system intended to be a solution to the problem. Thus, requirements on sensing for location recognition are discussed along with the specific sensing method chosen. A specific model/data representation is described, but only after a discussion of the needs that such a representation must fulfill. Consideration of the model update process illuminates key issues that are incorporated into the particular algorithm used. In each instance, I present an implementation that addresses the issues involved in the specific subproblem. The thesis, then, yields insight into the overall location recognition problem as well as an implemented solution to it.

1.4 Description of MARVEL

As stated, MARVEL is a system designed to recognize world locations via stereo vision based on models of those locations built by the system itself (see Figure 1-1). MARVEL obtains visual information about the world through a stereo vision input subsystem. This stereo data is abstracted into a form suitable for recognition. The

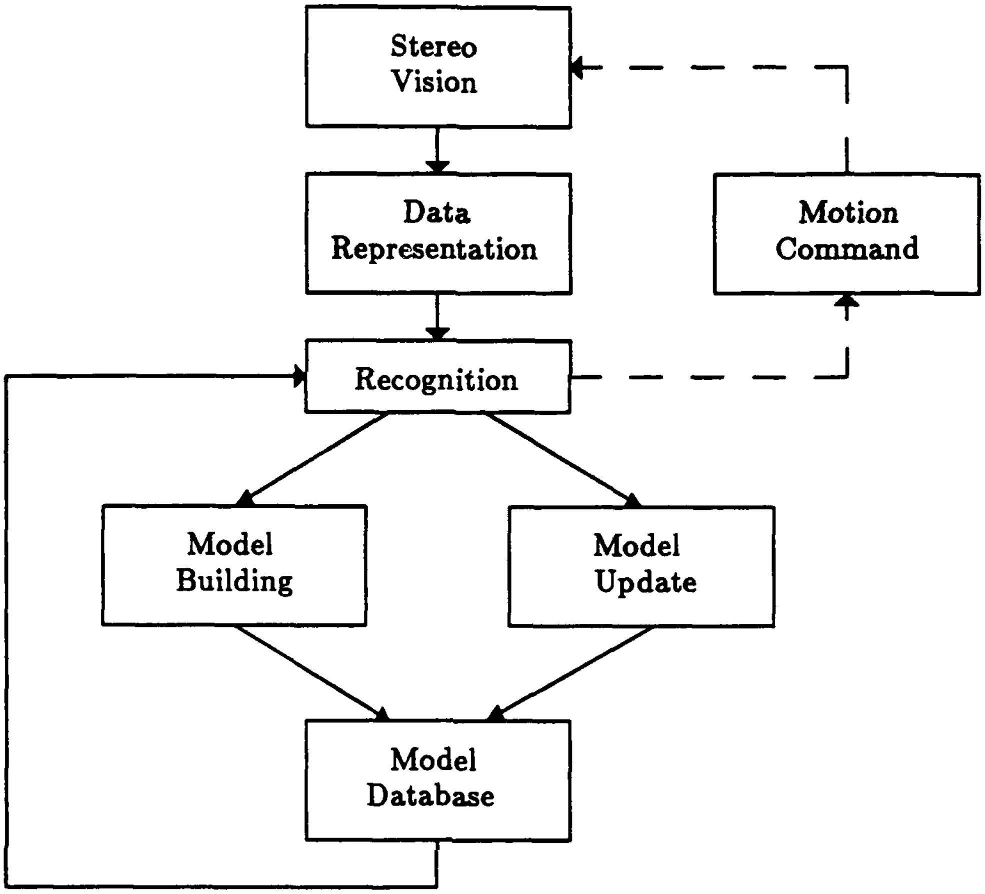


Figure 1-1: MARVEL block diagram.

recognition subsystem compares the current data with stored location models and decides if a match exists. If so, the data is combined with the matched model and stored back into the model database. The location corresponding to that matched model is then taken to be the current location. If no match is found, the data is stored in the model database as the model for a new location. If recognition does not succeed, but successful recognition is almost obtained, the robot can be commanded to move slightly to obtain a new view of the current location. This new data is then combined with the previous data and recognition proceeds against the combined data set.

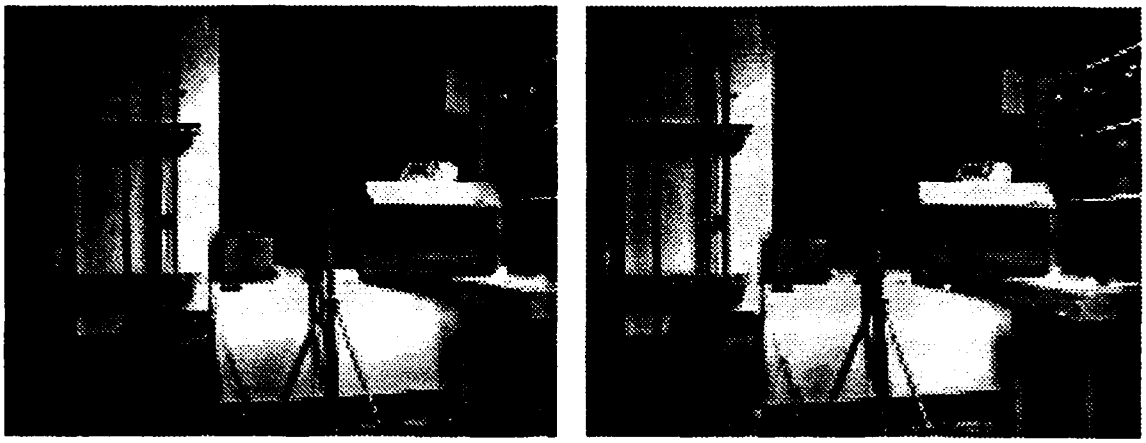


Figure 1-2: An image pair taken by MARVEL's stereo input system.

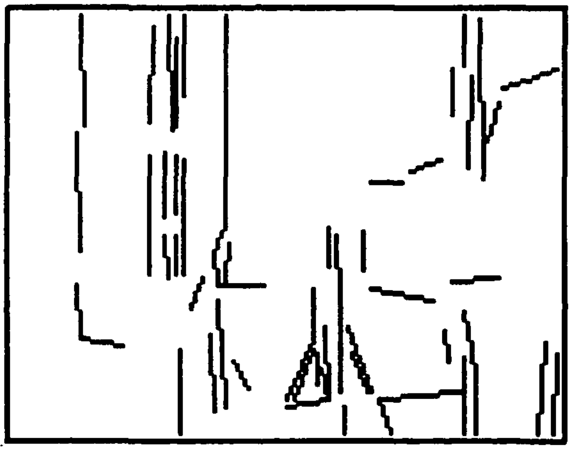


Figure 1-3: Matched stereo edges for the stereo pair in Figure 1-2. Each edge point is labeled with its 3-D coordinates by the stereo system.

1.4.1 Example of the Operation of MARVEL

Subsequent chapters describe MARVEL's various processing steps in depth. A brief example of the operation of MARVEL is presented here to provide context for the detailed examinations of the various subsystems.

MARVEL's stereo vision input system provides information about the world. A series of stereo image pairs is taken of the area surrounding the robot (e.g., Figure 1-2) and features are matched between the images of each pair to derive three-dimensional information about the surrounding environment (Figure 1-3). The 3-D information from each stereo pair is projected to the groundplane and together these projections form a 2-D representation of the currently visible location. An occupancy-grid data

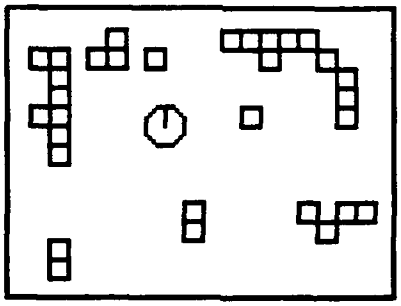


Figure 1-4: The 2-D grid representation of the stereo data derived from Figure 1-3 and from other views about the robot that have been projected to the groundplane.

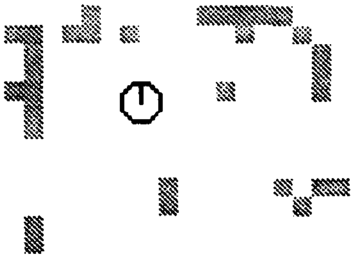


Figure 1-5: The initial location model, derived from the data shown in Figure 1-4.

abstraction is used to represent this 2-D information (Figure 1-4). The first data set obtained from each location is used to build the initial model for that location (Figure 1-5). The location models are improved over time as they are used for recognition. The improvements are reflected through the addition and deletion of modei points and through the adjustment of the weights of the model points. (The shading of a model point indicates its weight in the figures.)

To recognize the current location of the robot, MARVEL first obtains a set of data as described above (Figure 1-6). Based on the likely location of the robot, MARVEL then selects candidate models for recognition (Figure 1-7). The best alignment of model to data is found for each candidate (Figure 1-8) and the correct match is chosen (Figure 1-9). This recognition is used to update the model for future use (Figure 1-10). In the event that no valid match is found, the current location is determined to be a new location that the robot has encountered. In this case, a

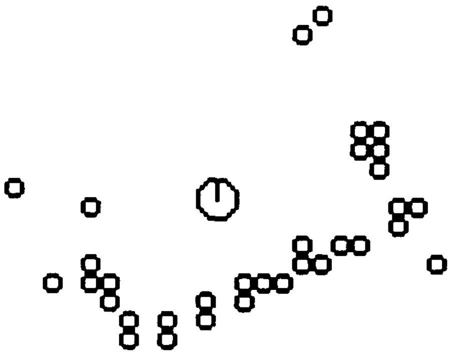


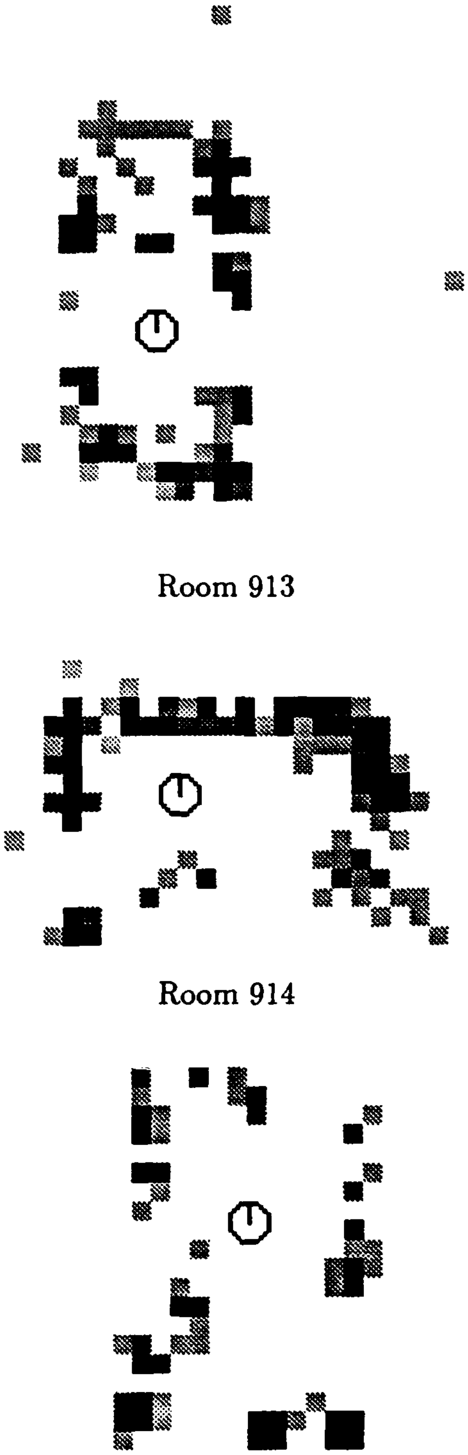
Figure 1-6: A data set to be recognized.

location model is created from the current data and entered . ıto the model database. Thus. MARVEL recognizes previously encountered locations and builds models for newly encountered locations for later recognition.

1.5 Summary of Results and Contributions

MARVEL provides a complete, working system for recognizing world locations to support navigation. The complete system operates autonomously, including the model building and updating of existing models. (This is in contrast to most existing recognition systems, which use handcrafted models.) Recognition proceeds using an aggregate set of data, no one part of which is sufficiently distinct to enable recognition independently. MARVEL responds to a changing world by changing its location models. MARVEL also deals with sensor and model error by modifying the location models.

To demonstrate the efficacy of MARVEL, extensive testing was performed and is reported herein. Over the course of more than 1000 trials, no false positives occurred while only a small number of false negatives were experienced. Good localization of the position of the robot in the recognized location was achieved in ‘his testing. The testing demonstrates the effectiveness of the ideas and methods described in this thesis.

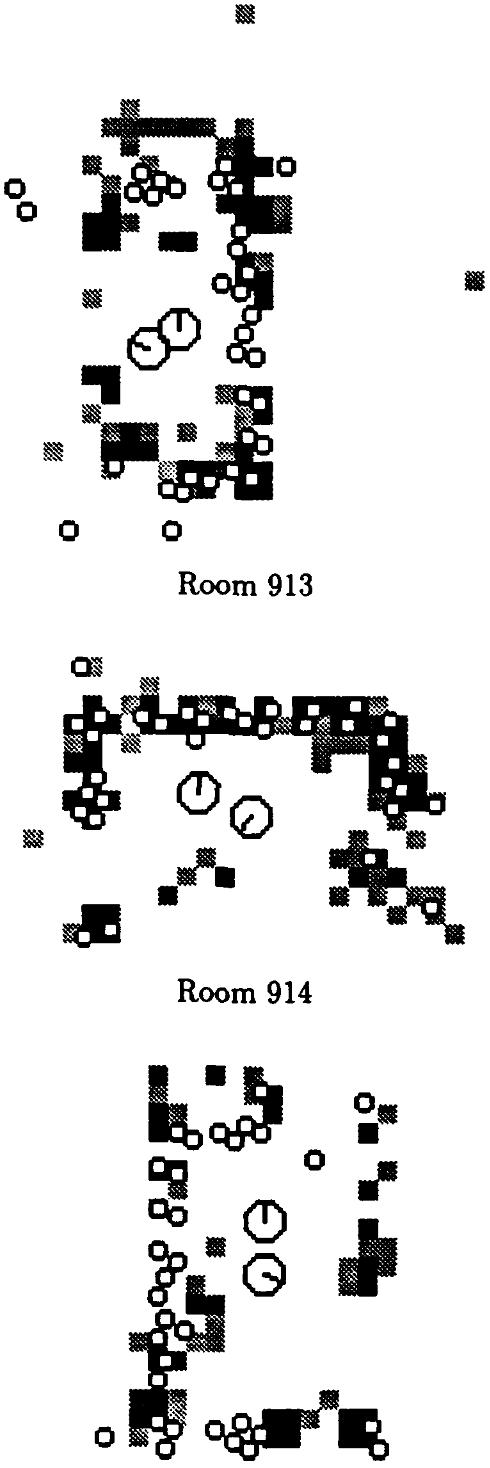


Room 913

Room 914

Room 915

Figure 1-7: Candidate models from the database for recognition of the data set shown in Figure 1-6.



Room 913

Room 914

Room 915

Figure 1-8: Recognition against the candidate models shown in Figure 1-7.

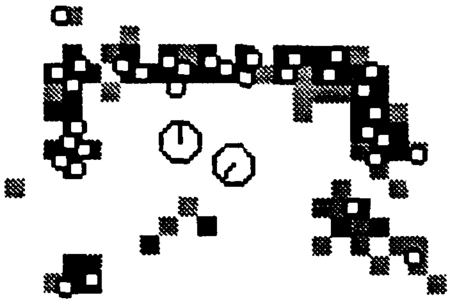


Figure 1-9: Correct recognition chosen from candidates in Figure 1-8.

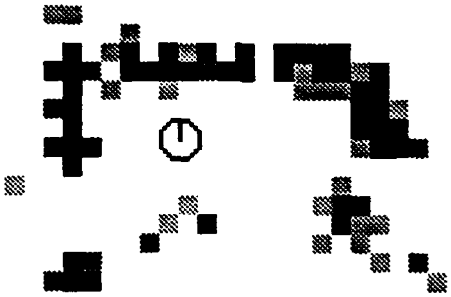


Figure 1-10: Updated location model based on the recognition result shown in Figure 1-9.

1.6 Overview of the Thesis

This thesis proceeds in much the same order as MARVEL operates. Chapter 2 discusses the issues of sensing for input data and describes a calibration procedure for the stereo hardware. Chapter 3 presents the stereo vision algorithm developed for use in MARVEL. From the stereo data, both the recognition data and the models are autonomously constructed by MARVEL. Chapter 4 describes the requirements on the data representation and shows how both the recognition data and models are derived from the stereo data input. Chapter 5 gives the details of recognition against a single location model. Any recognition algorithm should succeed against a single model; Chapter 6 describes how the fact that MARVEL is intended for use with a mobile robot can be used to advantage when performing recognition against a database of models. Since the models are built autonomously, the system must be able to change these models based on current information about the world. Chapter 7 describes the model update process, which is based on the recognition information. The testing methods and results for the recognition process are described in Chapter 8. MAR-

VEL's location recognition system fits into a larger world modeling system that is described in Chapter 9. The fact that a mobile robot is being used for the sensing platform opens up many possibilities for sensing and recognition. These strategies are discussed in Chapter 10. Finally, the lessons learned from MARVEL are summed up in Chapter 11, while Chapter 12 discusses future directions for research based on MARVEL.

Chapter 2

Stereo Input

In order to recognize locations, information characterizing these locations must be obtained. MARVEL uses a stereo vision system to obtain three-dimensional data from the local world surrounding the robot. The properties of such a vision system are well-understood and with proper calibration, stereo can be used to provide useful information about the environment.

2.1 Requirements on Sensing

Data obtained by the robot's sensors must have certain characteristics in order to be useful for recognizing the location of the robot. Chief among these are that the data characterize the location, have good localization, and be viewpoint independent.

If the recognition system is to have any hope of recognizing a location, its input data must characterize that location. Some quality of the individual pieces of data or of the data as a whole must be unique to the location from which it was derived so that a comparison can be made to a stored representation (model) of that location and a match determined. If a unique feature can be found and recognized for each location, then recognition of the various locations in the robot's world can proceed quite reliably. Such a method would be fragile, however, as any occlusion or sensor error that prevented the robot from sensing the distinguishing feature of a location would cause recognition to fail. Thus, it seems more advantageous to have *several* such distinguishing features per location so that the probability of seeing one of them is increased.

If several features are to be used to recognize a location, then each of these features must in turn be detected. For a complex feature, however, this detection presents yet another recognition problem in the general sense: the feature must be recognized in the input data and its position determined. A search procedure is also required to find the set of these features in the input reliably. Simple features can be detected much more easily and quickly than complex ones, but the price we pay for simplicity is a lack of uniqueness—the simpler the location feature, the less it helps to identify one unique location. With simple features, however, we can afford to find many of them in a location and use their aggregate to characterize the location. Since there are many of them, occlusions or input errors on a small number of them will not affect the recognition process greatly. We would expect that several different locations would have some of these simple features in common, but it would be rare that two locations would share the same aggregate of features. Using an aggregate of simple features to characterize a location, then, is potentially fast since the features would be easy to detect, yet robust due to the number of such features used to recognize any given location.

Location recognition is the main part of the problem that MARVEL is solving. An important subgoal, however, is the determination of the position of the robot in the location. For a floor-based mobile robot, this position is defined in terms of a two-dimensional translation (x and y) and rotation (θ) from a reference point in each location. Since position localization is based on the features that are used in the recognition process, such a localization can only be successful if the aggregate of the recognition features themselves is well-localized. There are two aspects to this requirement on the features. First, the recognition features must correspond to physical features that have well-defined, stable positions in the locations. Second, the determination of these feature positions by the input system must have a limited and well-understood uncertainty and the determination must also be repeatable. Only with these two requirements fulfilled can we hope to localize the position of the robot accurately.

Since the sensing platform is a mobile robot, the data for a given location can be obtained from almost any position within that location. The features used for recognition then, should be available to the sensors from most of the positions within a location. More importantly, the appearance of these features should not change as the position of the sensors changes. Without such viewpoint independence, the task of finding these features and using them for recognition is extremely difficult.

2.2 Advantages and Disadvantages of Stereo

The goal of stereo vision is the determination of 3-D coordinates of part or all of the visible world as presented in a pair of images taken from slightly different viewpoints. (For a somewhat dated, but comprehensive, review of the field, see [Binford 1982] or [Poggio and Poggio 1984].) The images are obtained from a pair of viewing positions with a known displacement. A typical arrangement is to have two cameras mounted and separated horizontally by a fixed distance. If a common point in the world can be determined between the two images obtained, then the relative positions and orientations of the cameras can be used to determine the 3-D coordinates of that point. ([Horn 1986] discusses the issues and mathematics of stereo matching and geometry.) Some stereo algorithms operate on regions (e.g., [Levine *et al.* 1973], [Ohta and Kanade 1985], [Witkin *et al.* 1986]) while others operate on visible, sparse features (e.g., [Moravec 1977], [Marr and Poggio 1979], [Grimson 1981], [Pollard *et al.* 1985a], [Prazdny 1985], [Ayache *et al.* 1985], [Drumheller and Poggio 1986]).

Stereo vision for the determination of 3-D coordinates of world points has well-known advantages and disadvantages. The uncertainties involved with 3-D coordinate determination have been studied (e.g., by [Torre *et al.* 1985] and [Matthies and Shafer 1986]). The information obtained from stereo vision typically has good angular resolution. The distance resolution, however, decreases with the distance of the matched features from the viewpoint. These uncertainties can be reduced by altering the stereo geometry, for example by increasing the baseline separation between the cameras or using lenses with long focal lengths, but there is a tradeoff. Such measures either increase the overall disparity range in a stereo pair, thus making the matching problem more difficult, or narrow the field of view, giving less coverage for the area of interest. It should be clear that stereo methods provide little information for regions that are visible in only one of the images. This occurs due to occlusion or to the slightly different viewing positions of the cameras. Depth information is determined most accurately for regions of the images with large variations in intensity and least accurately for homogeneous regions (due to the lack of information to determine the correct matches within these regions) [Kass 1988]. While comparisons can be made between the 3-D positions of objects or regions visible in a single stereo pair using uncalibrated stereo, calibration of the stereo setup is required to obtain accurate 3-D information. By evaluating the requirements on the input data for location recognition, we will see that stereo vision is a good choice for a sensing method for MARVEL.

2.2.1 Marr-Poggio-Grimson Stereo Algorithm

The stereo system used in MARVEL is based on the Marr-Poggio-Grimson Stereo Algorithm [Marr and Poggio 1979] [Grimson 1981] [Grimson 1985]. Details of this algorithm and the improvements made to it for MARVEL will be discussed in Section 3. This section considers that stereo algorithm with respect to the requirements on sensing discussed previously in Section 2.1.

Stereo Data

The Marr-Poggio-Grimson stereo algorithm provides 3-D information about the visible scene in terms of camera-centered world coordinates. World points that are visible in both the left and right images and that have been matched by the stereo algorithm are labeled with x-, y-, and z-coordinates, calibrated in feet. The position (0, 0, 0) is taken to be on the floor at the base of the robot. The x-axis extends in the positive direction to the right of the robot, the y-axis forward, and the z-axis upward.

Feature-Based

The Marr-Poggio-Grimson stereo algorithm is a feature-based algorithm, returning 3-D information at a set of features that are matched between the two images. These features are intensity edges, found by an edge detector that considers the zero-crossings of the images convolved with difference-of-Gaussian filters [Marr and Hildreth 1980], [Canny 1986], [Deriche 1990]. The resulting 3-D edges are easily detectable, relatively simple features that can be used for recognition.

Most intensity edges in the images correspond to physical edges or markings in the world. The 3-D edge information obtained from stereo, then, corresponds to stable features in the world. The existence and distribution of these features about the various world locations serve to identify them and are used by MARVEL for recognition. (Although image intensity edges can also correspond to features, such as shadow edges, that do not correspond to physical locations on surfaces, these unstable features are removed from the location models in the model update process—see Section 7.1.) The stereo output thus characterizes the world locations, a requirement mentioned earlier. (This will be explained further in Section 2.3.4.) Since the output of the stereo algorithm is based on physical features in the world, any object that is visible in both stereo images can potentially yield information that can be used to recognize the location being viewed. As required above, then, stereo data can be obtained independently of the viewpoint used to obtain it.

Sparse Data

As explained above, 3-D information is obtained by the stereo algorithm at only a sparse set of features visible in the images. This sparseness works to our advantage, however, since the stereo features only arise from world features that are sufficiently distinct to be detected and matched between the images. The distribution of these features about a location characterizes that location, since it is highly unlikely that two locations would give rise to sufficiently similar sets of features that confusion could result. The feature sets for two such locations would have to agree beyond the model thresholds used in recognition (Section 5.3) and the two locations would have to be close enough for both to be selected as candidate models from the database (Chapter 6) for this sort of confusion to occur. In order to demonstrate further the robustness of Marvel against just such an occurrence, very similar rooms were used for testing the system (Section 8.1).

Errors and Uncertainties

Any stereo algorithm has the possibility of returning data resulting from an incorrect match. One expects, however, the majority of the data from the algorithm to be correct. The Marr-Poggio-Grimson stereo algorithm (Section 3.1) along with enhancements added for this research (Section 3.4 Section 3.5) is relatively immune to this sort of mismatch error. The algorithm yields good localization of the location features, which was one of the requirements listed above. The localization uncertainties arise mainly from the use of discrete image information in the stereo matching process and, as mentioned earlier, there is a tradeoff of localization versus ease of matching. The uncertainties and errors in the implemented system are given in Section 2.3.3.

2.3 Stereo System Details

This section details the specifics of the stereo input system used during the testing of MARVEL. Exactly how these cameras were used to build the stereo input system and how the stereo image pairs were used to obtain the stereo data will be considered in terms of five distinct topics:

1. The geometry of the two-camera stereo configuration.

2. The specifications of the stereo configuration and of the cameras.

3. The errors and uncertainties inherent in stereo data.

4. The reliability of stereo data.

5. The method of obtaining 360° stereo information around the robot.

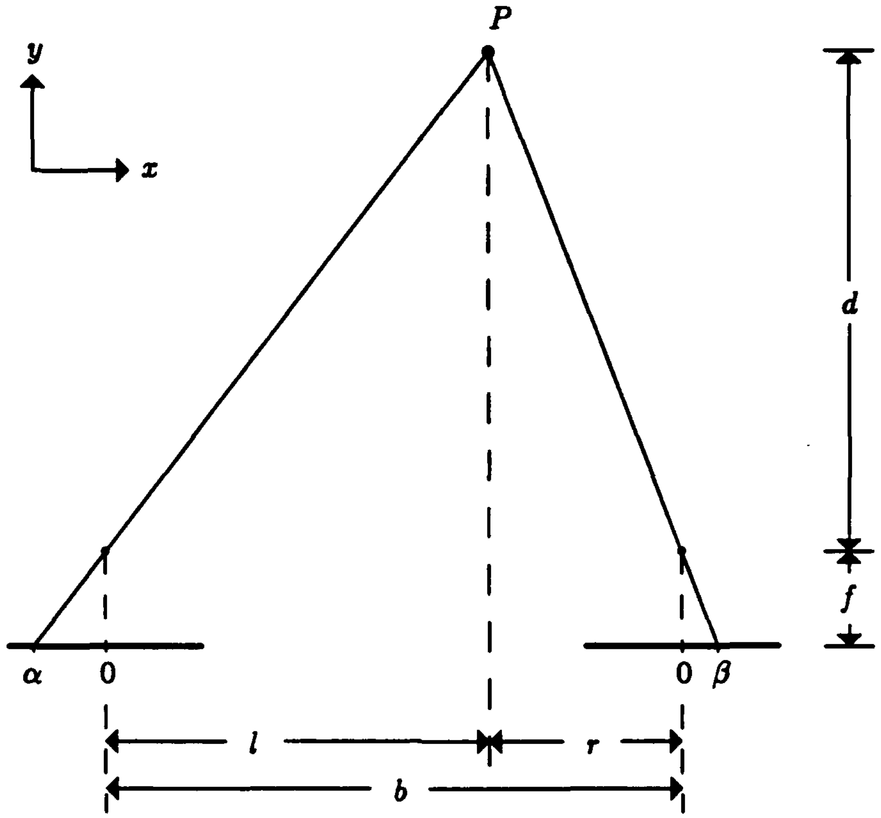


Figure 2-1: Stereo geometry (top view).

2.3.1 The Stereo Geometry

The cameras were aligned for a horizontal epipolar stereo geometry: the camera optical axes were parallel and the image planes were coplanar with no cyclotorsion (rotation offset about the optical axes). This geometry simplifies both the matching process (since any given world point P will appear in the same horizontal scan line in each image) and the conversion from image coordinates and disparity to 3-D world coordinates [Horn 1986]. The cameras were mounted so that the focal point of the left camera was directly over the axis of rotation of the robot. This axis of rotation, then, corresponded to the z-axis of the robot-centered world coordinate system. From this camera configuration (see Figure 2-1), the y-direction distance d in millimeters to a point P that is visible in both images is

$$d = \frac{fb}{w\delta} \tag{2.1}$$

where

f = the focal length of the camera lenses, in millimeters,

b = the baseline separation of the camera optical axes, in millimeters,

α = the horizontal pixel coordinate of the left-image point,

β = the horizontal pixel coordinate of the right-image point,

δ = $\beta - \alpha$, the horizontal disparity, in pixels,

w = the width of a pixel, in millimeters.

This distance d is the y-coordinate of the point P in the robot-centered world coordinate system. If the point P were at infinity, it would appear in the same position in the left and right images, i.e., $\alpha = \beta$. The disparity $\delta = \beta - \alpha$ is the displacement of the image of P in the right image with respect to the left image.

Once the distance d to the point P has been found, the x- and z-coordinates of the point can be determined for the world coordinate frame. The x-coordinate (in millimeters) is

$$x = \frac{w\alpha d}{f} \tag{2.2}$$

and the z-coordinate (in millimeters) is

$$z = \frac{h\gamma d}{f} + h_c \tag{2.3}$$

where

h = the height of a pixel, in millimeters,

γ = the vertical displacement of the left image point from the image cen'er, in pixels,

h_c = the height of the camera optical axis, in millimeters.

These calculations put the origin of the world coordinate system on the floorplane directly beneath the focal point of the left camera.

2.3.2 Stereo System Specifications

The stereo images were obtained using a pair of black-and-white Panasonic WV-CD50 CCD array cameras. The camera video signals were then digitized by a custom-built frame-grabber for processing on a Symbolics 3600 computer. Each camera was fitted with an 8.5 millimeter lens, which yielded a field of view of roughly 55°. The original images were 576×454 pixels with each pixel 0.0138×0.0134 millimeters. The images were downsampled by averaging each group of four pixels into one yielding an effective image size of 288×227 pixels and an effective pixel size of 0.0276×0.0268 millimeters. The pixel sizes were determined by a calibration procedure, which is discussed in Section 2.4. The baseline separation of the cameras was 4 inches. (Baseline separations less than roughly 3 inches could not be achieved due to the limitations of the camera mounts.) For testing, the cameras were mounted $44\frac{5}{8}$ inches off of the floor.

2.3.3 Errors and Uncertainties

Errors in the output of a stereo algorithm are due to incorrect matches between the left and right images. Although steps are taken to avoid such mismatches (see, for example, Sections 3.4 and 3.5), no stereo algorithm is perfect and errors do occur. During the testing of MARVEL, over 650 stereo pairs of real data were taken and matched. Less than 5% of these stereo pairs gave rise to matching errors. For the pairs where errors did occur, less than 1% of the matched points were matched incorrectly.

Perfect image data cannot be obtained from the cameras due to optical system imperfections. These optical distortions, however, typically affect the projected location of a point in an image by less than one pixel [Lenz and Tsai 1987]. Discretization of the images also affects the input data, preventing an exact determination of the locations of the intensity edges used by the stereo algorithm. For example, if the horizontal location of an edge point in the left image is determined to be x, then the actual edge point could lie anywhere between $x+0.5$ and $x-0.5$ and still be labeled at position x. The matching right-image edge point has this same uncertainty, yielding

x-coordinate			y-coordinate			z-coordinate		
true	min	max	true	min	max	true	min	max
0.167	0.150	0.183	10.000	9.111	11.081	3.719	3.701	3.733
0.167	0.142	0.191	15.000	13.085	17.572	3.719	3.691	3.739
0.167	0.134	0.199	20.000	16.734	24.850	3.719	3.680	3.745

Table 2.1: Stereo uncertainties due to discretization (in feet).

a disparity uncertainty of ±1 pixel. This disparity uncertainty gives rise to an uncertainty in the distance determined for a point, which is its y-coordinate (Equation 2.1), and thus also leads to an uncertainty in its x- and z-coordinates (Equations 2.2 and 2.3). For the stereo geometry described above, these uncertainties are summarized in Table 2.1 for a world feature point centered between the two cameras at various distances from the robot. (The uncertainties can be lowered by interpolating the positions of the zero-crossings to achieve sub-pixel localization of them before matching [MacVicar-Whelan and Binford 1981].)

2.3.4 Reliable Data

The stereo data obtained from a pair of images corresponds to intensity edges in those images. Most of these intensity edges are caused by the edges of objects in the world. These edges characterize tne location in which they are found because they correspond to objects in that location. The recognition algorithm uses both the existence of these features and their distribution in the location.

The positions of small objects in a room, such as cups and books, tend to change often. Large objects, such as whiteboards and bookcases, do not move very often. This observation leads to an important heuristic for these types of scenes:

> **Heuristic**: Large objects tend to have stable positions. Therefore, the existence and distribution of these objects in a location may serve to identify that location.

Also, a known property of stereo algorithms is that features that extend perpendicularly to the baseline of the cameras are most easily and accurately matched. This fact, along with the heuristic just mentioned, indicates that certain stereo edges will be more useful for recognition than others. The edges that most accurately characterize a location are the ones that are long and vertical.

The actual requirement for long, vertical edges that is enforced in MARVEL is not overly strict. Edges must be within 25° of vertical. In practice, straight lines are fit to the 3-D stereo data using a method given by [Pavlidis 1982]. The slopes of these 3-D lines are checked to determine if the edges are sufficiently vertical. The roughly vertical edges that are longer than 2 feet are then used as the representative data for the current location.

The fact that stereo supplies accurate 3-D data combined with the domain assumptions allows MARVEL to perform some preprocessing on the vertical edges obtained. Since some floors are reflective (e.g., tile floors), some matched stereo features may correspond to reflections of objects in the room. The z-coordinates of the reflection features will be negative, however, due to their apparent positions beneath the floorplane. Any such negative-z features are eliminated from the data set. By checking their heights, ceiling features can also be eliminated. Such features are typically lighting fixtures that cover a large part of the ceiling area. Due to their locations on the ceiling, these features are not reliably in the field of view of the cameras and therefore cannot be counted on for recognition.

The stereo data that is used to represent a location is reliable in two different senses. First, because the edges used are roughly vertical, the stereo matching and the resulting 3-D position determination for them can be performed accurately. Second, because the edges are relatively long, they correspond to significant objects in the location. The length helps ensure that a correct stereo match is found, since the match must occur over a large number of edge points. Finally, such long vertical edges are highly unlikely to be noise edges in the images.

2.3.5 360° View

The field of view encompassed by a stereo pair is roughly 55°. Considering that parts of the left image are not visible in the right image, and vice versa, one stereo pair covers about a 50° field of view, which represents slightly less than 1/7 of the available information surrounding the robot. Since the recognition algorithm is relying on the distribution of a set of simple features, more data must be obtained by rotating the cameras to see the other parts of the location. By rotating about the origin of the robot-centered world coordinate system, information from different viewing directions can easily be combined. Since the stereo geometry has been arranged so that the z-axis of the robot-centered world coordinate system passes through the focal point of the left camera (see Section 2.3.1), the camera pair should be rotated about the vertical line passing through the left camera focal point. The question, then, is how to obtain 360° coverage from successive stereo pairs. Two different possibilities exist for obtaining this additional data.

The best solution for obtaining the additional stereo pairs is to have the cameras mounted on a motor-driven turntable. Such a turntable driven by a stepper motor can achieve better than a 2° accuracy, while a D.C. motor with an optical shaft encoder for position feedback can achieve better than 1° accuracy [Flynn 1990]. A set of eight stereo pairs can then be taken at 45° rotation to obtain full coverage of the current location.

If the robot can turn in place (e.g., the robot described by [Brooks 1986]), then the additional views of the current location can also be obtained by turning the robot itself. The shaft encoders on the robot wheels supply information about how far the robot has turned, but there will be some slip between the wheels and the ground. Also, such a robot typically turns in a very tight circle. (The robot just mentioned inscribes a 1 centimeter diameter circle while turning in place.) Thus, the rotation information available from the robot is not extremely accurate. In this case, smaller rotations should be used between stereo pairs so that overlapping views are obtained. The overlaps can then be compared to determine the exact rotation between views.

MARVEL was tested using the second method of obtaining stereo pairs. The stereo cameras were rotated by 20° between each stereo pair, yielding 18 stereo pairs per data set for each location. More than half of a given image, then, overlaps with the one from the previous rotation. This overlap region is used to determine the rotation between the stereo pairs.

The camera configuration has been arranged so that the focal point of the left camera is at the center of rotation. The effect of a rotation can thus be approximated by a simple translation of features in the left image. Therefore, the 2-D edges in the left images from successive stereo pairs are matched to determine the translational offset between the images. This offset is then used to determine the actual rotation of the stereo cameras. The amount of rotation between views can be increased for sufficiently complex scenes while still retaining enough common features between the views to allow the actual rotation to be found.

The actual matching between successive stereo pairs is performed on the significant vertical edges found in each left image. The matches are determined by considering the intervals between these edge, using a scheme similar to that of [Ohta and Kanade 1985]: the match between the edges of the two images is chosen that yields the most consistent pairings of the intervals. This scheme allows for edges that might be missing due to input errors. Because the motion of the features is translational, the matching process is relatively straightforward and the rotations can be reliably verified.

2.4 Calibration

From the above discussion, it should be apparent that three different types of calibration are needed to obtain reliable information from the stereo system. First, the cameras need to be adjusted so that their optical axes are parallel to the floorplane. Second, the cameras must be turned to achieve the epipolar stereo geometry. Finally, the image centers and the pixel size of the cameras used must be determined in order to convert from image coordinates to world coordinates.

Setting the camera optical axes parallel to the floor is relatively simple. A marker is set at some distance from the cameras at the same height as the camera lens centers and the cameras individually tilted up or down as needed to bring the mark to the image centers. Note that these optical image centers need to be determined for each

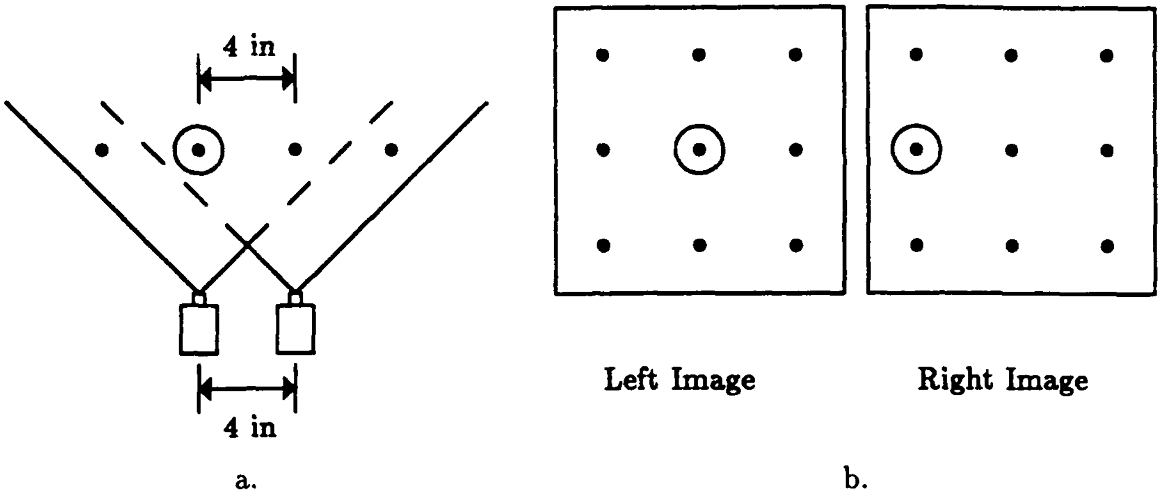


Figure 2-2: Setup for epipolar optical axis calibration. a. Top view. b. Camera images.

camera/lens pair.

The epipolar geometry requires that the optical axes be parallel and the image planes be coplanar. A 4 × 5 foot planar test pattern consisting of a set of 1 inch diameter black dots arranged in a square grid with 4 inch spacing was constructed. (Recall that the baseline between the cameras was also set to 4 inches.) The cameras were set roughly parallel at a distance so that the test pattern filled the left image. The left camera was then turned until the spacing between the dots on the left side of the image matched the spacing between the dots on the right side. This sets the left-camera image plane parallel to the test pattern. The right camera was then turned so that, when its image was shifted to align its image center with the left image center, the dots in the left and right images overlapped, but were off by one phase. That is, each dot in the right image that coincided with a dot in the left image was actually a dot on the test pattern lying 4 inches to the right of the corresponding dot seen in the left image (see Figure 2-2). The optical axes are then parallel since they are the same distance apart both at the cameras and at the test pattern. After this procedure, the image planes of the cameras are within a few millimeters of being coplanar, which suffices.

The remaining task, then, is to determine the apparent pixel size in the images and the image centers. The calibration procedure used follows the methods of [Lenz and Tsai 1987]. (Other interesting and useful work on camera calibration can be found in [Tsai 1986], [Tsai and Lenz 1987], and [Tsai and Lenz 1988].) By pixel size, we actually mean the horizontal and vertical spacing between pixels, but it is more convenient to think in terms of pixel width and height. In typical CCD array cameras, the discrete information from each row of camera sensor elements is converted to an analog signal and these analog signals from each row are sequentially combined into an analog video signal. This analog signal is then sampled by an A/D converter and used to form a discrete image array that is displayed on a CRT screen and used for processing. It is evident, then, that the height of an image pixel is the same as the height of a CCD camera sensor element. The width of an image pixel is determined by comparing the clock frequency of the camera to the sampling frequency of the A/D converter:

$$w_p = w_e \frac{f_c}{f_{A/D}}$$

where

w_p = the width of an image pixel, in millimeters
(actually the horizontal distance between successive pixels),

w_e = the width of an camera sensor element, in millimeters
(actually the horizontal distance between successive elements),

f_c = the sensor clock frequency of the camera,

$f_{A/D}$ = the sampling frequency of the A/D converter.

For the camera and frame grabber described in Section 2.3, we have $w_e = 0.0172$ millimeters, $f_c = 9.615$ MHz, and $F_{A/D} = 12$ MHz, yielding a pixel width of 0.0138 millimeters. The apparent pixel size is then 0.0138 × 0.0134 millimeters (width × height) for a full 576 × 454 pixel image.

The image centers were found using the Radial Alignment Constraint described by [Lenz and Tsai 1987]. The method used is outlined here. Interested readers should refer to [Lenz and Tsai 1987] for full details.

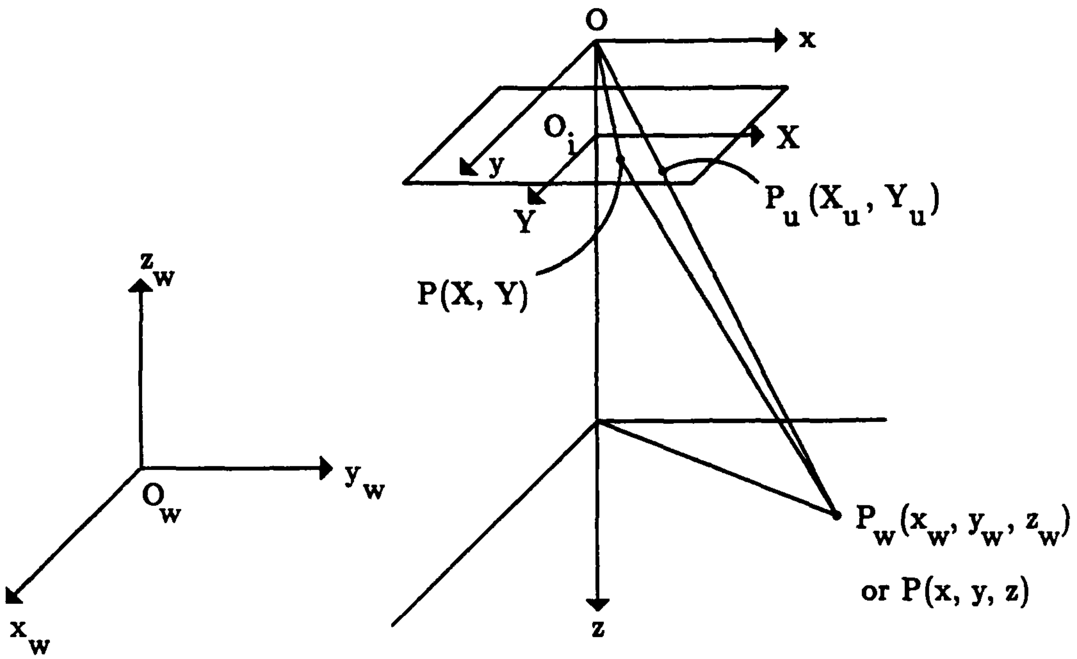


Figure 2-3: Camera imaging geometry with perspective projection and radial lens distortion.

Figure 2-3 shows the camera imaging geometry with perspective projection and radial lens distortion. The world coordinate system is shown at the left, with origin O_w. The camera 3-D coordinate system is based at O and the image center is at O_i. Object point P_w with world coordinates (x_w, y_w, z_w), if projected through a perfect pinhole camera, would appear at point P_u with image coordinates (X_u, Y_u). Due to radial lens distortion, however, it actually appears in the image as point P at (X, Y).

The overall transformation from world point (x_w, y_w, z_w) to image point (X, Y) can be viewed as having four parts [Lenz and Tsai 1987]:

1. A rigid body transformation from world coordinates to camera coordinates:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.4}$$

2. Perspective projection:

$$X_u = f\frac{x}{z} \qquad Y_u = f\frac{y}{z} \tag{2.5}$$

3. Radial lens distortion, modeled with a second-order polynomial:

$$X = X_u (1 + \kappa R^2)^{-1} \qquad Y = Y_u (1 + \kappa R^2)^{-1} \tag{2.6}$$

with

$$R^2 = X_u^2 + Y_u^2.$$

The determination of the radial lens distortion is not necessary for our purposes. The fact that it occurs, however, is used to formulate the Radial Alignment Constraint that is exploited to determine the image centers (see below).

4. Camera sensor coordinate to image pixel coordinate conversion.

This is the determination of image pixel sizes and that was discussed earlier.

Note that in Figure 2-3 the vectors (x, y), (X_u, Y_u), and (X, Y) are radially a'igned when the image center is correct. This alignment holds independently of any radial lens distortion, the focal length of the lens, and the z-component of the translation (t_z in Equation 2.4). This fact forms the basis of the Radial Alignment Constraint (RAC). The RAC can be stated as

$$(x, y) \parallel (X_u, Y_u) \parallel (X, Y). \tag{2.7}$$

It follows from Equations 2.7 and 2.4 that

$$\frac{X}{Y} = \frac{x}{y} = \frac{x_w r_1 + y_w r_2 + z_w r_3 + t_x}{x_w r_4 + y_w r_5 + z_w r_6 + t_y}. \tag{2.8}$$

The dot test pattern described earlier provides a set of coplanar points at known world coordinates $P_{wi} = (x_{wi}, y_{wi}, z_{wi})$. By choosing a world coordinate system where $z_{wi} \equiv 0$, Equation 2.8 yields a linear form of the RAC:

$$Y_i x_{wi} r_1 + Y_i y_{wi} r_2 + Y_i t_x - X_i x_{wi} r_4 - X_i y_{wi} r_5 - X_i t_y = 0. \tag{2.9}$$

Five or more calibration points yields an overdetermined set of homogeneous equations, allowing the unknowns $(r_1, r_2, t_x, r_4, r_5, t_y)$ to be determined up to a common scaling factor. Dividing through by t_y, we obtain

$$Y_i x_{wi}\frac{r_1}{t_y} + Y_i y_{wi}\frac{r_2}{t_y} + Y_i\frac{t_x}{t_y} - X_i x_{wi}\frac{r_4}{t_y} - X_i y_{wi}\frac{r_5}{t_y} - X_i = 0. \tag{2.10}$$

An estimate $\hat{X}_i$ of the x-coordinate of the point (X_i, Y_i) that is dependent on these unknowns can then be obtained from Equation 2.10:

$$\hat{X}_i = Y_i \frac{x_{wi}\frac{r_1}{t_y} + y_{wi}\frac{r_2}{t_y} + \frac{t_x}{t_y}}{x_{wi}\frac{r_4}{t_y} + y_{wi}\frac{r_5}{t_y} + 1}. \tag{2.11}$$

Using a least-squares minimization, Equation 2.11 can be solved for $(\frac{r_1}{t_y}, \frac{r_2}{t_y}, \frac{t_x}{t_y}, \frac{r_4}{t_y}, \frac{r_5}{t_y})$ by minimizing the error $X_i - \hat{X}_i$. Now, the point image coordinates (X_i, Y_i) are measured from some estimate of the optical center of the image (C_X, C_Y), which is taken as the image origin. If we change our notation for an image point (X_i, Y_i) to $(X_i + \Delta C_x, Y_i + \Delta C_y)$, we can rewrite Equation 2.10 as

$$\begin{aligned} Y\,(x_{wi}\frac{r_1}{t_y} + y_{wi}\frac{r_2}{t_y} + \frac{t_x}{t_y}) - X\,(x_{wi}\frac{r_4}{t_y} + y_{wi}\frac{r_5}{t_y} + 1) \\ = -\Delta C_x\,(x_{wi}\frac{r_4}{t_y} + y_{wi}\frac{r_5}{t_y} + 1) + \Delta C_y\,(x_{wi}\frac{r_1}{t_y} + y_{wi}\frac{r_2}{t_y} + \frac{t_x}{t_y}), \end{aligned} \tag{2.12}$$

where the unknowns are $(\Delta C_x, \Delta C_y)$. Using the same set of points, we again have an overdetermined set of equations that can be solved via least squares, thus improving the estimate of the image origin (optical center) (C_X, C_Y). This process is iteratively repeated, first solving Equation 2.10 using the new image origin and then Equation 2.12, until the change in image origin $(\Delta C_x, \Delta C_y)$ becomes small.

Using the Radial Alignment Constraint method just described, the image centers were determined for the images of the left and right camera-lens pairs. After less than ten iterations, the image centers were changing by only a fraction of a pixel. The final image centers determined for the full 576 × 454 pixel images were (317.33, 226.00) for the left image and (306.33, 223.67) for the right.

2.5 Summary

We have seen how the task of location recognition leads to certain requirements on the sensing system used for MARVEL. Stereo vision has suitable properties for use as MARVEL's input system. The essential characteristics of the chosen stereo algorithm (the Marr-Poggio-Grimson stereo algorithm) were discussed. The vision hardware used for the stereo input was described, along with a calibration procedure that permits accurate 3-D data to be obtained. The implementation of the stereo algorithm used for MARVEL is explained in the next chapter. This implementation takes a new approach to stereo feature matching using the Disparity Space representation. New strategies for validating correct matches and selecting among ambiguous matches have also been implemented.

Chapter 3

Stereo Matching in Disparity Space

The Marr-Poggio-Grimson stereo algorithm has been presented as a model of the human stereo vision process [Marr and Poggio 1979] [Grimson 1981]. The algorithm was subsequently revised by [Grimson 1985] to improve its performance with no claims as to the relevance of these revisions to human stereo vision. Proceeding from this work, we present herein further modifications to the algorithm used to implement this theory of stereo vision. These modifications involve a change of representation that makes the constraints of the stereo matching problem more accessible and facilitates experimentation with methods of feature matching [Braunegg 1989a] [Braunegg 1990].

In the Marr-Poggio-Grimson stereo algorithm, the features that are matched between left and right images are zero-crossings of the images convolved with difference-of-Gaussian filters of a range of sizes. The matching is performed across a range of eye fixation positions, with matches in a small area around each fixation position compressed into a single match that is then further processed. The method of feature matching presented here, rather than processing matches about individual fixation points, finds unique matches at each fixation point and then considers the set of matches across the full range of fixation positions.

3.1 The Marr-Poggio-Grimson Stereo Algorithm

The Marr-Poggio-Grimson stereo algorithm as implemented by Grimson consists of eig^ht steps, outlined below. The steps pertinent to the disparity-space matching algorithm will be explained more fully in the following sections, while a detailed explanation of the complete algorithm can be found in [Grimson 1985].

0. *Loop Over Levels:* Iterate from coarser to finer levels of representation. The level of representation is reflected by the value of the filter width w for the convolution step.

   1. *Convolutions:* Convolve the left and right images with $\nabla^2 G(w)$ filters.

   2. *Zero-Crossings:* Locate the non-trivial zero-crossings in the convolved images and mark these zero-crossings with their contrast signs.

   3. *Loop Over Fixation Position:* Iterate over a range of alignments of the left and right images.

      4. *Matching:* Identify valid feature matches between the left and right images.

         a. *Feature-Point Matching:* Given a disparity δ_0, match positive zero-crossings against positive ones and negative zero-crossings against negative ones over a vertical range of $\pm\epsilon$ and a horizontal range of $\pm w$ about the current alignment.

         b. *Figural Continuity:* Compress the contours matched about δ_0 from 4.a. into a single representation and eliminate those matched contours whose vertical lengths are less than a threshold.

         c. *Disparity Updating:* For the remaining contours matched about δ_0 record in the disparity map for the current level the average disparity of the matched contour points.

      5. *Loop:* Loop to Step 3 and repeat for all possible image alignments.

   6. *Disambiguation:* Use zero-crossing matches from coarser channels to disambiguate matches at the current level.

7. *Loop:* Loop to Step 0 and repeat for next finer level of representation.

8. *Consistency:* Eliminate zero-crossing matches that are inconsistent with coarser channel matches.

We propose a modification to the above algorithm at Steps 4 and 5. In the modified version, feature points are not matched in a range about each fixation position, but rather are matched only at the fixation positions. Figural continuity is then applied to the matched contours as they lie across the full range of disparities (fixations) considered. (A figural continuity test was later added to Grimson's implementation as well [Grimson 1989].) Specifically, the new steps of the algorithm are

4. *Matching:* Identify valid feature matches between the left and right images.

   a. *Feature-Point Matching:* Given a disparity δ_0, match positive zero-crossings against positive ones and negative zero-crossings against negative ones over a vertical range of $\pm\epsilon$ about the current alignment. The $\pm\epsilon$ is used solely to allow for a (possible) slight vertical misalignment between the images.

   b. *Disparity Updating:* For the contours matched at δ_0 record in the disparity map for the current level the disparity of the matched contour points.

5. *Loop:* Loop to Step 3 and repeat for all possible image alignments.

5-1. *Contour Following:* Form linked lists of matched contour points in the disparity map for the current level. These linked lists are the candidate matched contours for their corresponding left-image contours.

5-2. *Contour Validity Checking:* Test the matched contours in the current level's disparity map for validity.

   a. *Horizontal Segment Extension:* Connect matched contour segments across regions of horizontal contour points where the horizontal extensions have a disparity gradient less than a threshold, h.

b. *Figural Continuity:* Eliminate those matched contours whose vertical lengths are less than a threshold, f. ***The contours are followed without regard to how they vary in disparity.***

c. *Disparity Gradient:* Eliminate segments of matched contours that vary in disparity by an amount greater than a threshold, g. Recheck figural continuity for any contour from which a segment is removed.

5-3. *Matched Contour Disambiguation:* Disambiguate contour points with two or more candidate matches.

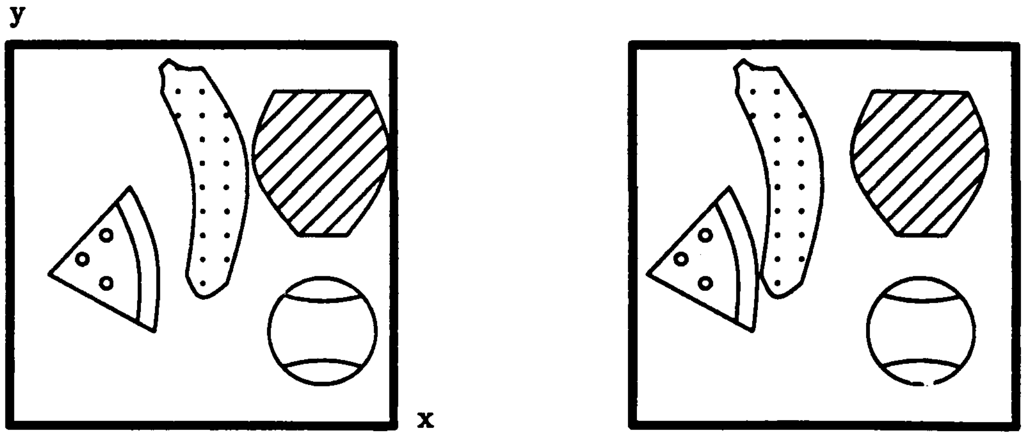
a. *Contour Subsumption:* Eliminate those contour matches that are subsumed by other matches.

b. *Consistent Disparities:* Where two or more matches remain for a contour point, choose the disparity most consistent with the disparities found for the rest of the contour.

The changes made to the algorithm will be discussed in the following sections. At this point we simply note that the differences lie in how the contours are matched and how the matched contours are checked for validity. The validity checks listed above are modifications of tests from the original algorithm while the disambiguation tests are new.

3.2 Disparity Space Contours

The new steps in the stereo matching algorithm involve a change in representation. In the new algorithm, matched contours are represented as contours in *disparity space*. *Disparity space* is a three-dimensional space: the x- and y-dimensions are the same as in one of the images (typically the left image) (Figure 3-1) while the third dimension is disparity. Matched contours are plotted as sets of ordered points in this space. Thus, for any matched pair of contour points (Figure 3-2), the matched point is explicitly represented in terms of its position in the (left) image and its perceived disparity (Figure 3-3). If we were to convert the disparity measurements



Left Image **Right Image**

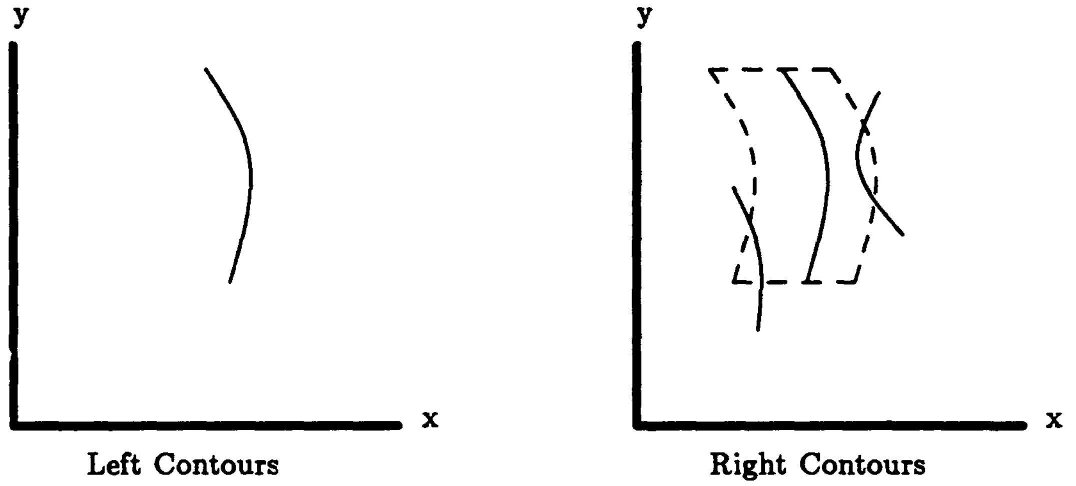
Figure 3-1: Left and right images.

to distance from the cameras (using information about the camera geometry), the disparity space representation would become a scaled model of the arrangement of the physical contours in the three-dimensional world.

When performing operations on the matches of individual left-image contours, we do not need to deal with them in the full three-dimensional disparity space. Since we know where a contour lies in the x and y dimensions (the left image location of the contour), we can consider the matches of this contour in a two-dimensional plane—the first dimension being distance along the contour and the second disparity (Figure 3-4). This *disparity-space plane* is a two-dimensional plane embedded in the three-dimensional disparity space.

It follows that points of the contour matches are referenced by arc length (based at the beginning of the contour) and disparity rather than x-coordinate, y-coordinate, and disparity. Thus, three-dimensional operations on contour matches in disparity space are reduced to two-dimensional operations in the disparity-space planes.

The disparity space representation for matched contour points has several benefits. By considering a matched contour as a whole, this new representation aids in the selection of breakpoints for long contours (Section 3.3.1) and the extension of matches across horizontal contour segments (Section 3.4.1). (Long contours often exist in a difference-of-Gaussian filtered image due to the requirement that zero-crossings form closed contours, rather than to the existence of corresponding real-world object



Left Contours

Right Contours

Figure 3-2: Left- and right-image contours. A single left-image zero-crossing contour is shown. For the right image the area of possible matches (in dotted lines) for the left-image contour as disparity varies is shown. Also shown are the right-image contours that are candidate matches for the chosen left-image contour.

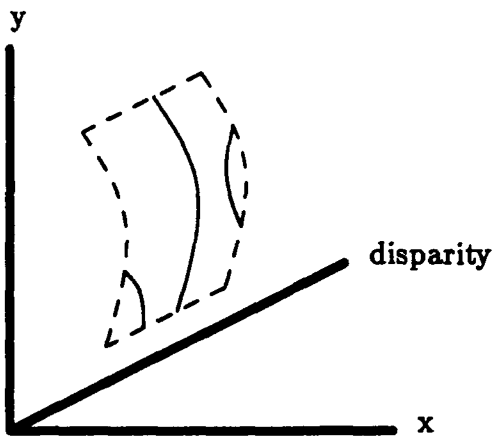


Figure 3-3: Disparity space, showing the candidate matches for the left-image contour that appears in the dotted box of Figure 3-2.

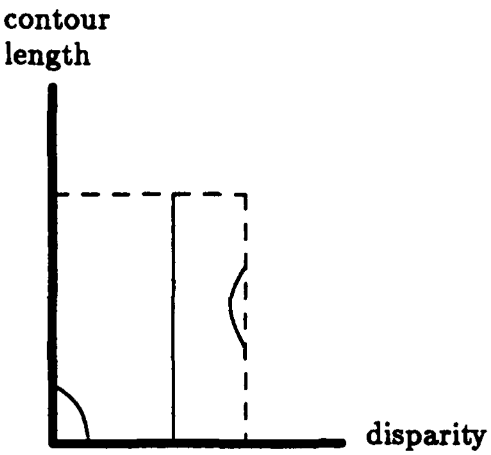


Figure 3-4: Disparity-space plane for the left-image contour of Figure 3-2, showing the candidate matches. This is the two-dimensional plane embedded in the three-dimensional disparity space shown in Figure 3-3.

features.) The representation allows us to implement a cleaner form of the figural continuity constraint (Section 3.4.2). It facilitates the application of a disparity gradient constraint to the whole matched contour or any part thereof (Section 3.4.3). Finally, the disparity space representation allows us to check easily for consistent disparities and contour subsumption when disambiguating candidate contour matches (Sections 3.5.1 and 3.5.2).

3.3 Matching to Yield Disparity Space Contours

Following from [Grimson 1985], we assume that the convolved left and right images,

$$\begin{aligned} LC_w(x,y) &= \nabla^2 G(w) * L \\ RC_w(x,y) &= \nabla^2 G(w) * R \end{aligned}$$

have been computed. $\nabla^2 G(w)$ denotes the Laplacian of a Gaussian whose central negative portion has width w, $*$ is the convolution operator, and L and R denote the left and right images, respectively. For each of these convolved images, the nontrivial zero-crossings have been located and marked with their contrast signs, yielding the bit maps:

$$\begin{aligned}
LP_w(x,y) &= \text{positive zero-crossings of } LC_w(x,y)\\
LN_w(x,y) &= \text{negative zero-crossings of } LC_w(x,y)\\
LH_w(x,y) &= \text{horizontal zero-crossings of } LC_w(x,y)\\
RP_w(x,y) &= \text{positive zero-crossings of } RC_w(x,y)\\
RN_w(x,y) &= \text{negative zero-crossings of } RC_w(x,y)\\
RH_w(x,y) &= \text{horizontal zero-crossings of } RC_w(x,y).
\end{aligned}$$

The zero crossing points $LC_w(x,y)$ and $RC_w(x,y)$ can also be grouped according to the edges, or contours, from which they arise. These left- and right-image contours are denoted by $LE_{w,k}(i)$ and $RE_{w,k}(i)$, respectively, where the contours are numbered $k = 1,2,\ldots$ in each image and the points along a contour are numbered $i = 1,2,\ldots$. Now, to find the candidate right-image matches for the k^{th} left-image contour $LE_{w_0,k}$ at the current filter width w_0, we construct a disparity-space plane

$$P_{w_0,k}(\delta,i) = \begin{cases} RP_{w_0}(X,Y) & \text{for } LE_{w_0,k}(i) \in LP_{w_0}\\ RN_{w_0}(X,Y) & \text{for } LE_{w_0,k}(i) \in LN_{w_0}\\ RH_{w_0}(X,Y) & \text{for } LE_{w_0,k}(i) \in LH_{w_0}\\ 0 & \text{elsewhere} \end{cases}$$

over

$$\delta_{min} \leq \delta \leq \delta_{max} \quad \text{(disparity range)},$$
$$0 \leq i < \text{length}(LE_{w_0,k}),$$

where

$X = X(E(w_0,k,i,\delta))$ is the x-coordinate of $E(w_0,k,i,\delta)$,

$Y = Y(E(w_0,k,i,\delta))$ is the y-coordinate of $E(w_0,k,i,\delta)$,

$E(w_0,k,i,\delta)$ is the point in the left image plane δ pixels to the right along the epipolar line through $LE_{w_0,k}(i)$,

$LE_{w_0,k}(i)$ is the i^{th} point in contour $LE_{w_0,k}$.

The disparity space plane for a left-image contour thus represents the possible matches of its points with points in right-image contours, described in terms of the disparity of a match versus the position of the matched left-image contour point along the contour (Figure 3-4).

In the case of horizontal epipolar lines, matching the points of horizontal contours is ambiguous. For such points, we rely on the Horizontal Segment Extension of Step 5-2.a for determination of the disparities. The disparity-space planes then become

$$P_{w_0,k}(\delta,i) = \begin{cases} RP_{w_0}(X,Y) & \text{for } LE_{w_0,k}(i) \in LP_{w_0} \\ RN_{w_0}(X,Y) & \text{for } LE_{w_0,k}(i) \in LN_{w_0}. \\ 0 & \text{for } LE_{w_0,k}(i) \in LH_{w_0} \end{cases}$$

Certain computational advantages are realized by implementing the disparity space contour representation for matching zero-crossings instead of the $\pm w$ window method of [Grimson 1985]. At a given disparity δ_0, only one point in the right image must be compared with a contour point in the left image. Grimson's method entails combining $2w + 1$ points from the right image into a single data point and comparing this data point with a contour point in the left image. If we assume horizontal epipolar lines in the images, the fact that we only match points at a single horizontal disparity allows us to search for matching points by simply scanning across the right image ov the range of horizontal disparities (see Figure 3-2). No additional processing is required to consider $\pm w$ windows around each fixation position. (We note that in both algorithms the $\pm\epsilon$ vertical range is achieved by overlaying copies of the right image at the required vertical offsets and using the resulting image for matching.)

3.3.1 Contour Following in Disparity-Space Planes

Once we have formed the disparity-space plane for a left-image contour, the candidate matching contours from the right image are easily found (Step 5-1). We simply scan the rows of the disparity-space plane until we find a matched point. We then trace matched points along the contour starting from this matched point, forming a linked list of matched points. Given matched point $P_{w_0,k}(\delta_0, i_0)$, we search for the next matched point in the range $[P_{w_o,k}(\lfloor \delta_0 - c \rfloor, i_0 + 1),\ P_{w_o,k}(\lceil \delta_0 + c \rceil, i_0 + 1)]$ where c is a threshold on the maximum jump in disparity between successive contour points (larger than that allowed by the disparity gradient discussed in Section 3.4.3). All of the candidate matching contours in the disparity-space plane are found in this way, possibly breaking the original contour into several matched contour pieces. We then check these candidate matches for validity and disambiguate when multiple matches have been found for the left-image contour.

As stated in Step 4.a, we consider candidate matches over a vertical range of $\pm\epsilon$ about the epipolar line along which we expect to find matches. Because of this, the matched points in a disparity-space plane may form lines that are more than one pixel wide. We perform a thinning operation on such lines to obtain one pixel wide candidate matching contours in the disparity-space plane. Rather than using a standard "grass fire" approach to thinning, we bias the thinning algorithm to prefer lines that vary least in disparity over their lengths. Due to this thinning operation and the fact that the y-axis of the disparity-space plane represents arc length along the contour, the resulting candidate matches in the plane form lines that are strictly monotonic in the vertical axis of the plane.

3.4 Matched Contour Validity Checking in Disparity Space

After obtaining the candidate matching right-image contours for each left-image contour, we apply several validity tests to each of the candidate contours in disparity space. These tests are applied to the candidate contours individually and must be applied to every candidate contour. We cannot assume that a candidate match is valid simply because it is a unique match for a particular left-image contour.

3.4.1 Horizontal Segment Extension

Contour matching cannot be applied to horizontal contour segments when we have horizontal epipolar lines. If we do not consider horizontal points, however, we may unintentionally break a long matched contour segment into two or more shorter contour segments. To avoid this problem, we extend contour segments across horizontal sections where possible (Step 5-2.a). Suppose that a matched contour segment in a disparity-space plane ends at the point $P_{w_o,k}(\delta_0, i_0)$. We search the points in the range $[P_{w_o,k}(\lfloor \delta_0 - jh \rfloor, i_0 + j),\ P_{w_o,k}(\lceil \delta_0 + jh \rceil, i_0 + j)]$ for $j = 1, 2, 3, \ldots, \text{length}(LE_{w_0,k})$ until we find a matched contour point. (h is the threshold mentioned in Step 5-2.a of the modified algorithm.) If no point is found, we cannot extend the contour. If a point is found, say at $j = j_n$, then the points

$$P_{w_0,k}(\delta_0 + 1, i_0 + 1),\ P_{w_0,k}(\delta_0 + 2, i_0 + 2),\ \ldots\ ,\ P_{w_0,k}(\delta_0 + j_n, i_0 + j_n)$$

are added to the contour and we continue to trace the contour in the disparity-space plane from the point $P_{w_0,k}(\delta_0 + j_n, i_0 + j_n)$.

Typically, the threshold h is set to the same value as the threshold c mentioned in Section 3.3.1 above. The threshold h is the maximum allowable jump in disparity between contour points where one or both of the points comes from a horizontal section of the contour.

Horizontal segment extension is facilitated by the disparity space representation of candidate matches (Figure 3-5). To extend a match across a horizontal section of a contour, we simply search for the next y-coordinate (in the direction of increasing arc length) until matched points are found. Choose the matched point at this y-coordinate whose x-coordinate (disparity) is closest to that of the original point. If any y-coordinates that correspond to non-horizontal contour points are crossed, the extension fails. If we have extended the contour for n points and the difference in disparity across the extension does not exceed nh, then we accept the extension and connect the matches across that horizontal contour section.

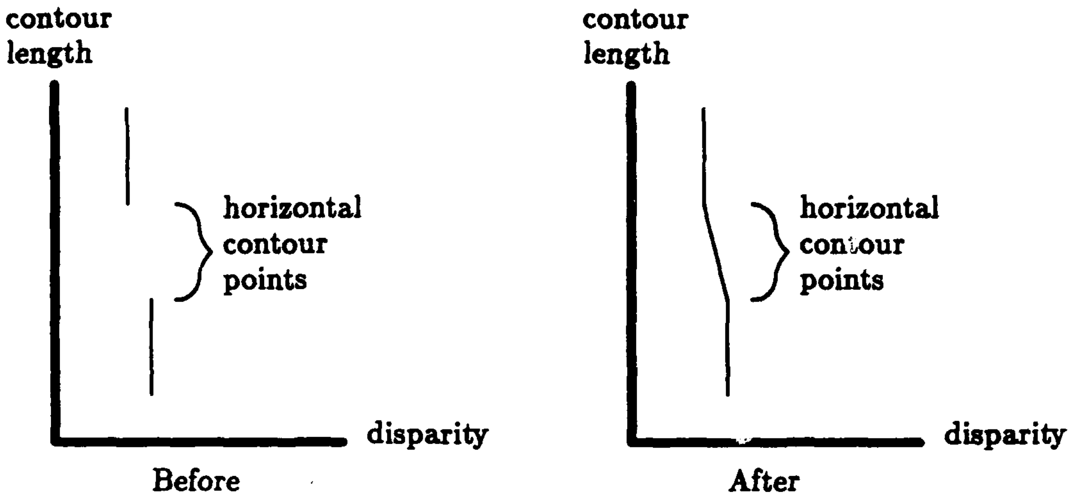


Figure 3-5: Contour in a disparity-space plane before and after horizontal segment extension.

3.4.2 The Figural Continuity Constraint

Arguments based on both the cohesiveness of matter and psychophysical evidence support the use of figural continuity as a verification criterion for stereo matches [Mayhew and Frisby 1980] [Mayhew and Frisby 1981], [Prazdny 1985]. In the implementation of the figural continuity constraint given by [Grimson 1985], one pixel wide gaps were allowed in a contour when checking its length. After experimenting with the stereo algorithm, however, we have found that this allowable gap feature is rarely used. (Grimson also found this to be true [Grimson 1989].) Therefore, we have implemented a simplified figural continuity constraint based on the continuous length of a matched contour (Step 5-2.b). To apply the figural continuity test, we simply eliminate all matched contours that are vertically shorter than f pixels long.[1] Contour points from horizontal segments are not counted in the length of a contour because they are interpolated between matched contour points instead of resulting from matched points themselves.

Note that the length of the contour is determined regardless of how the contour varies in disparity. The Disparity Gradient test (see the following section) explicitly considers how a contour varies in disparity. The disparity space representation of a

[1] The threshold f is the *figural continuity length* and is typically expressed as a fraction of the width of the central region of the $\nabla^2 G$ filter.

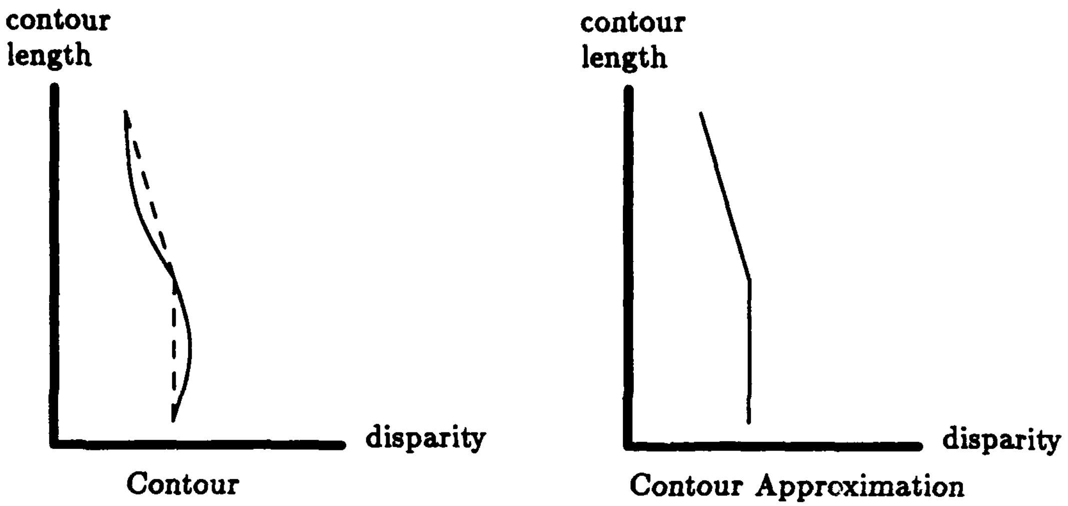
matched contour allows the figural continuity constraint to be applied to the contour as a whole. In the original algorithm, the length of a matched contour can only be checked in a window of width $2w$ pixels of disparity. As presented by Mayhew and Frisby, figural continuity should be applied to a matched contour as a whole, not just to arbitrary pieces of it. With the disparity space representation for matched contours, we have separated the length requirement for matched contours from consideration of how widely they range in disparity.

3.4.3 The Disparity Gradient Constraint

Psychophysical observations have shown the importance of a disparity gradient limit for stereopsis [Burt and Julesz 1980a], [Burt and Julesz 1980b]. These observations were the basis of the PMF stereo algorithm [Pollard *et al.* 1985a], [Pollard *et al.* 1985b]. Only a weak condition on the disparity gradient, however, is incorporated in the original Marr-Poggio-Grimson stereo algorithm. Below we provide an explicit check of the disparity gradient using the disparity-space representation of contour matches.

Disparity gradient for a matched contour is directly interpreted from the slope of the matched contour in the disparity-space plane for the contour. (The x-axis corresponds to disparity and the y-axis to arc length along the contour.) Since the discrete nature of the representation makes defining and finding gradients difficult, we define a procedure for finding the gradient of a matched contour over various sections of the contour (Step 5-2.c). Rather than choose a fixed segment length over which to calculate the gradient, we let the segment length vary.

To apply the Disparity Gradient constraint to a matched contour, we first obtain a straight-line approximation [Pavlidis 1982] to the contour in the disparity-space plane, including the horizontal segment extensions, if any (Figure 3-6). We let points deviate from these straight-line segments by two pixels (typically) to allow for positional errors due to discretization and the $\nabla^2 G$ convolution. Segments with disparity gradient greater than g are deleted, while the remaining segments are accepted as valid contour matches by this processing step.



Contour

Contour Approximation

Figure 3-6: Straight-line approximation to a matched contour in a disparity-space plane.

It is important to note that Step 5-2.c (Disparity Gradient) of the new algorithm was implicit in Step 4.a of the original algorithm. In the original algorithm, a matched contour was not permitted to vary by more than $2w$ pixels across the figural continuity length. It could vary quite rapidly within these bounds (and thus contain large disparity gradients), however, and still be accepted. The new algorithm places a specific limit on how quickly a contour may vary in disparity (i.e., a disparity gradient limit). This disparity gradient test is distinct from the figural continuity test and also does not affect the matching process since we match contours at a particular horizontal disparity instead of over a range of disparities.

The fact that we explicitly check the disparity gradient along a matched contour allows us to break the contour into distinct segments, eliminating parts of the contour whose disparity gradients are too steep. This situation often occurs on zero-crossing contours in $\nabla^2 G$ images since these contours must be closed. Many times strong zero-crossing contour segments are closed by weak zero-crossings that do not correspond to physical features in the real world. Due to the nature of these weak zero-crossing segments, they tend to wander in the image, producing matched zero-crossing segments that vary in disparity. The varying disparity of these weak contour segments is typically evidenced by high disparity gradients, allowing many of these segments to be located and eliminated.

Typically, the threshold g is smaller than the thresholds c and h used in Steps 5-1 and 5-2.a (see Sections 4.1 and 5.1 above). This is due to the fact that a contour can vary sharply in disparity over a few points but still have a low disparity gradient overall.

3.5 Matched Contour Disambiguation

After validating the matched contours using the figural continuity and disparity gradient criteria, ambiguous matches often remain. We attempt to resolve these ambiguities by checking for contour subsumption and consistent disparities along contours. Afterward, any remaining ambiguities are resolved, if possible, by considering nearby contour matches from the current and coarser filter channels (as in the original Marr-Poggio-Grimson stereo algorithm).

3.5.1 Contour Subsumption

When two or more matches are possible for a given contour, one of the matches frequently subsumes the others. That is, one candidate match extends over a larger portion of the contour and the portion that it does cover includes some of the other matches. If this is the case, we accept the subsuming match as the correct one (Step 5-3.a). This is justified because the base of support for that match is stronger (hence the longer length of the match) than for the others.

The disparity space representation of contour matches facilitates direct comparison of the extents of matches of consistent disparities (Figure 3-7). The starting and ending points of the matched portions of contours are directly represented in disparity space. In our implementation of contour subsumption disambiguation, we initially require strict subsumption, i.e., the subsuming contour must completely cover the range of the subsumed contour as well as extend above and below it. This requirement is relaxed if the subsuming contour extends well beyond one end of the subsumed contour but falls a few pixels short of the other end.

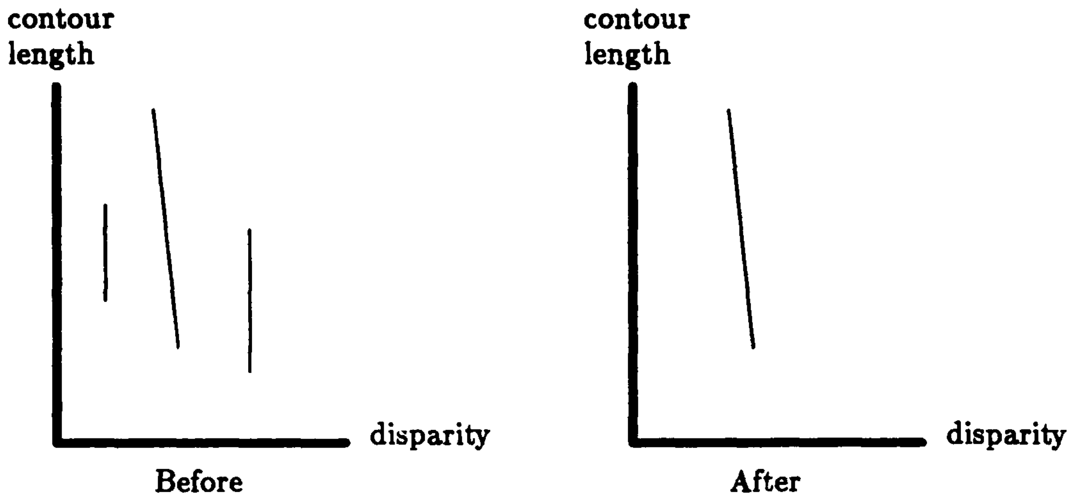


Figure 3-7: Contours in a disparity-space plane before and after subsumed contours are removed.

Thus, the candidate match

$$P_{w,k}(\delta_i, i),\ P_{w,k}(\delta_{i+1}, i+1),\ \ldots,\ P_{w,k}(\delta_{i+m}, i+m)$$

for left-image contour k subsumes candidate match

$$P'_{w,k}(\delta_j, j),\ P'_{w,k}(\delta_{j+1}, j+1),\ \ldots,\ P'_{w,k}(\delta_{j+n}, j+n)$$

if $m > n$ and either

$$i < j \quad \text{and} \quad (i+m) > (j+n),$$

$$i > j \quad \text{and} \quad (m-n) > \xi \geq (i-j),$$

or

$$(i+m) < (j+n) \quad \text{and} \quad (m-n) > \xi \geq [(j+n) - (i+m)],$$

where ξ is small.

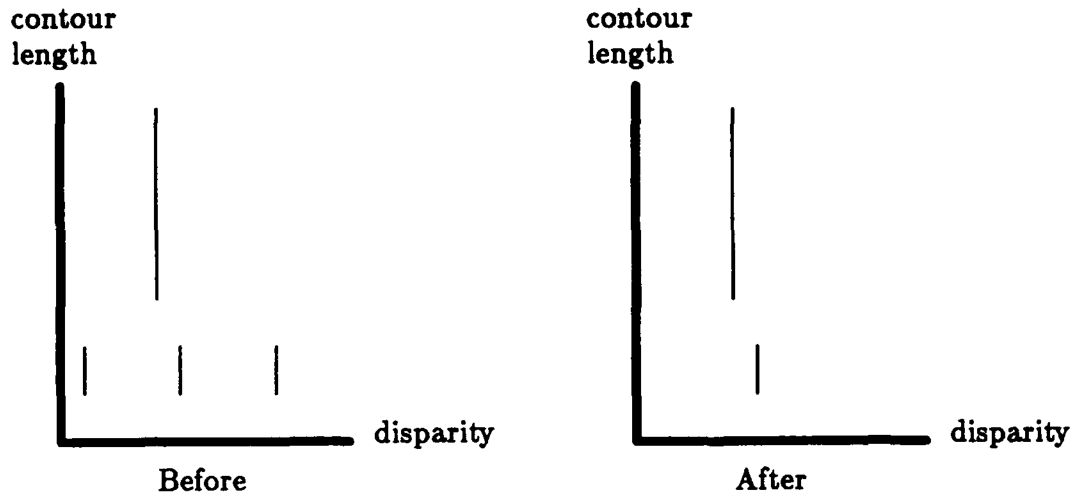


Figure 3-8: Contours in a disparity-space plane before and after ambiguous matches with inconsistent disparities are eliminated.

3.5.2 Consistent Disparities

To disambiguate remaining contour points with more than one match, we choose the match whose disparity is most consistent with the disparities found for the rest of the contour (Step 5-3.b) (Figure 3-8). Consider a contour point with matches at disparities δ_1, δ_2, ..., δ_n. Let N_{δ_i} be the number of points from the same contour that are unambiguously matched at disparity δ_i. Assign disparity δ_j to the contour point when δ_j maximizes N_{δ_j} for $1 \leq j \leq n$ and $N_{\delta_j} > 0$. If all of the $N_{\delta_j} = 0$, i.e., there are no unambiguous contour points whose disparity matches one of the possible disparities of the point in question, then select the match whose disparity is closest to an unambiguous disparity for the contour: choose δ_j such that δ_j minimizes $|\delta_j - \delta_i|$ for $1 \leq j \leq n$ over all i. If this minimum distance exceeds a threshold, i.e., $\min |\delta_j - \delta_i| > \epsilon$, $\forall i, j$, however, then the ambiguously matched point is left unmatched at this stage. These remaining ambiguous points will be disambiguated by using matched contour information from coarser channels (Step 6).

3.6 Advantages of the Disparity-Space Approach

The use of the disparity-space plane representation for candidate contour matches has several advantages over the origiııal Marr-Poggio-Grimson implementation. These advantages are based on the representation of arc length versus disparity for the candidate matches of a contour.

With the disparity-space plane representation for the contour matches, the figural continuity is directly observable from the candidate matches. Using this representation, the figural continuity constraint is applied once per matched contour. The original implementation required this constraint to be checked for the matches found in the $\pm w$ range at each fixation position. Even assuming that a matched contour lies at a single disparity, the original method would check the figural continuity of this contour $2w + 1$ times.

To check the disparity gradient of a matched contour, we must follow the contour in the x- and y-coordinates of the original image as well as in disparity. By representing contour matches in disparity-space planes, however, we consider the matched contours in a two-dimensional space: one dimension is disparity while the second is distance along the contour. Thus contour points are referenced by disparity and arc length (based at the beginning of the contour) rather than disparity, x-, and y-coordinates (Figure 3-4). We can therefore compare candidates for zero-crossing matches in these disparity-space planes instead of in a three-dimensional space.

Contour subsumption is easily determined in the disparity-space planes. The linked lists of points that comprise a match for a contour are parameterized by arc length, allowing direct comparison of different candidate matches.

Finally, consistency of disparity along a contour can be enforced naturally using the disparity-space plane representation. This follows from the fact that the matched contour points are parameterized by disparity in the representation.

3.7 Conclusions

By considering matched contours in disparity space, we are able to improve the Marr-Poggio-Grimson in several ways. The figural continuity constraint can be applied to complete matched contours instead of sections of them that are bounded by a fixed disparity range. An explicit disparity gradient threshold can be applied to matched contours and sections of the contours that do not meet this constraint can be removed. Computational savings are realized by eliminating the horizontal disparity window formerly used in the matching process (Step 4.a), both by allowing a simpler matching process and by permitting us to apply the figural continuity constraint only once per matched contour.

Disambiguation of competing contour matches is facilitated by the disparity-space plane representation of the matches. The explicit representation of arc length makes contour subsumption easy to determine. The explicit representation of disparity along the contour matches allows us to check the consistency of the disparity of candidate matches in a straightforward manner.

With this new disparity-space plane representation for stereo matching, the task of the matching problem is evident—the best matches appear as long, connected lines in the disparity-space plane. The explicit representation of arc length and disparity of the candidate matches aids validation and disambiguation of the matches. Finally, the unnecessary detail of the actual location of the contours is eliminated for the matching, validation, and disambiguation operations.

3.8 A Disparity-Space Stereo Matching Example

Although the simplified figures shown in this chapter help one to understand the disparity-space representation and operations, it is interesting to see the effects of these operations on a real stereo image pair. Unfortunateiy, the limits of black-and-white reproduction make it difficult to present the depth results of the stereo algorithm, so only the matched contour points themselves can be shown. Figure 3-9

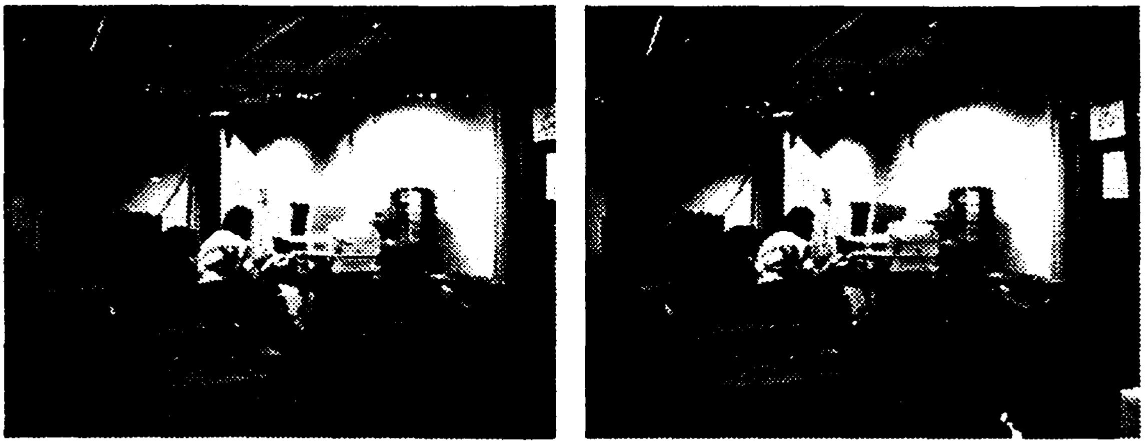


Figure 3-9: The left and right images of a stereo pair.

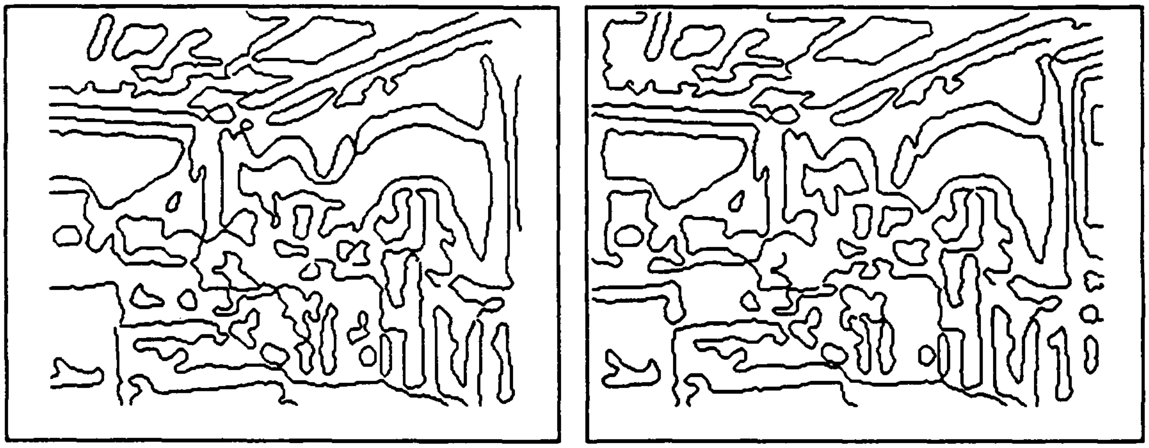


Figure 3-10: Zero-crossing contours from the left and right images of Figure 3-9.

shows a stereo image pair. The zero-crossing contours found for each image are shown in Figure 3-10. These contours are matched by the stereo algorithm as described in this chapter. Figures 3-11 through 3-16 show the various stages of matching, validation, and disambiguation for these image contours, starting with the unambiguously matched left contour points in Figure 3-11, i.e., those points that have a unique match among the right contour points. The later stages of processing will determine which of these matches are correct. Figure 3-16 shows the final set of matched left contour points, each of which are assigned 3-D coordinates by the stereo algorithm.

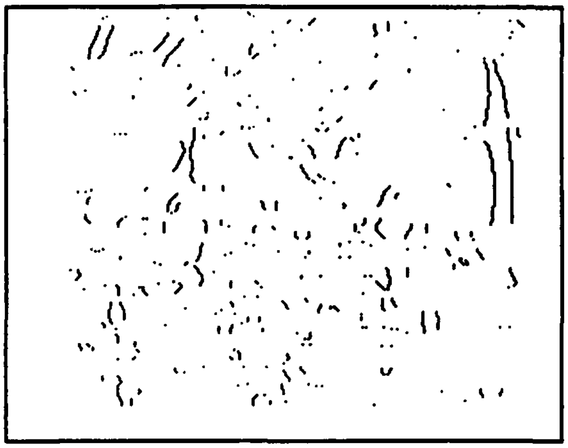


Figure 3-11: Original unambiguously matched left contour points from Figure 3-10 (Step 4).

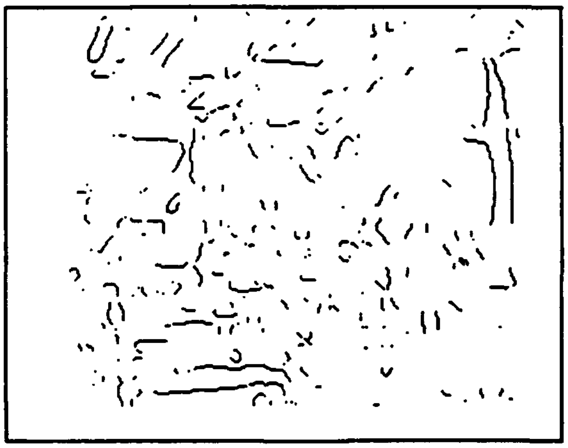


Figure 3-12: Unambiguous left contour matches after horizontal segment extension (Step 5-2.a).

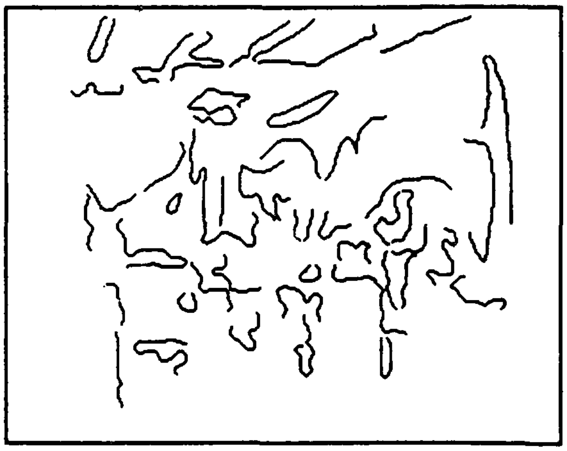


Figure 3-13: Left contour matches after imposing figural continuity constraint (Step 5-2.b).

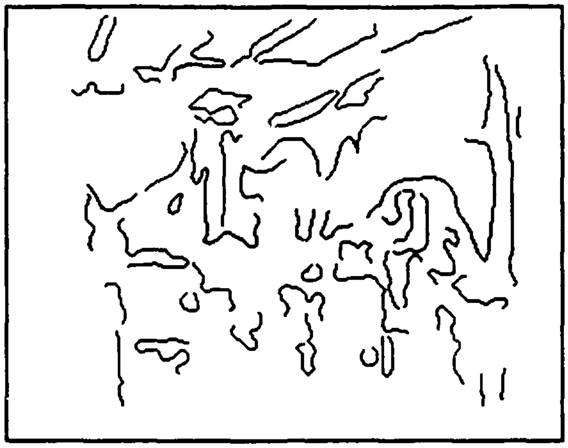


Figure 3-14: Left contour matches after imposing disparity gradient constraint (Step 5-2.c).

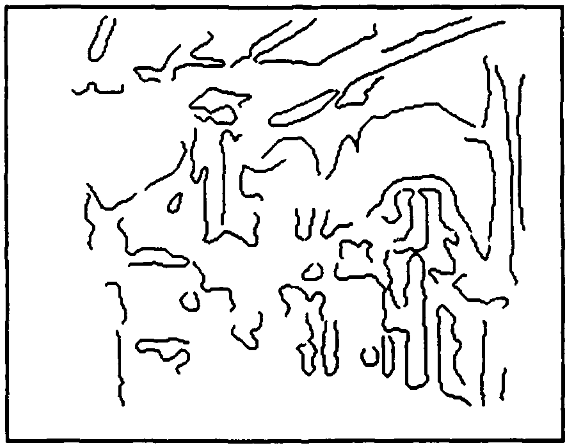


Figure 3-15: Left contour matches after eliminating subsumed contours (Step 5-3.a).

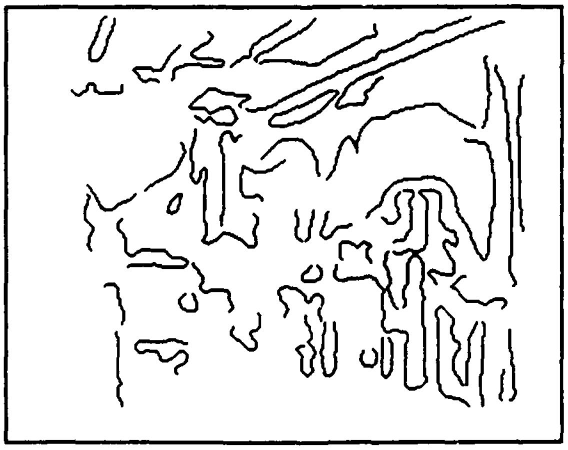


Figure 3-16: Left contour matches after eliminating inconsistent disparities (Step 5-3.b).

3.9 Summary

This chapter described a novel approach to matching stereo features through the use of the *disparity space* representation. This representation was used in the framework of the Marr-Poggio-Grimson stereo algorithm. Matching contours in disparity space is straightforward because the information needed for matching is made explicit in this space. Contour validity checking and disambiguation are also easily accomplished through the use of this representation. An example with real data was given demonstrating the use of the disparity space methods described. The 3-D information obtained from this stereo algorithm is used to represent the locations to be recognized, as described in the next chapter.

Chapter 4

The World Is Defined by What We See

It would be wonderful if the sensors available to a mobile robot could give detailed, blueprint-like floorplans of the rooms it encounters. Unfortunately, no such sensors exist. Given that the output of the stereo vision system chosen for MARVEL consists of vertica! edge features from the current location labeled with 3-D coordinates, how can this data be used to represent the world? Specifically, the data must permit the robot to build location models for the rooms it encounters and must also be amenable to comparison with these models so that the identity of the visible room can be determined.

4.1 The Task Is Recognition, Not Navigation

In developing any representation, the first question to ask is, "What task is to be performed using this data?" Different representations are suitable for d.fferent tasks, and a good representaticn should make explicit the constraints that the data applies to the task. As described in Section 2.1, the aggregate of a large set of simple features characterizes a location. Simple features are easily detectable so that effort can be expended where it belongs—on recognizing the location and not on recognizing the individual features. By using an aggregate of such features, the dependence on any one or small number of them is reduced, making the system robust to occlusion. The stereo system provides just such an aggregate of simple features: a set of 3-D line segments distributed in space, whose distribution characterizes the location to be recognized.

One goal of a representation is to provide an abstraction of the raw data from the input. The abstraction should retain essential information for the task while hiding or eliminating details that are not needed. Through testing MARVEL it became apparent that 2-D information suffices for accomplishing recognition of rooms in indoor environments. Other researchers, most notably the group at INRIA (*Insitut National de Recherche en Informatique et en Automatique*) in France, have been using 3-D stereo data for a mobile robot to determine freespace and identify objects in the vicinity of the robot [Boissonnat *et al.* 1988a, 1988b] [Le Bras-Mehlman *et al.* 1988] [Faugeras *et al.* 1986]. Their work is concerned with planning safe paths through the immediate environment and not with determining where the robot is in a global sense. The position determination that they accomplish is aimed primarily at relating the current position to the one where a previous data set was obtained. Note, however, that other researchers in the field, for example [Wells 1989] and [Kriegman *et al.* 1989], have come to the same conclusion about world representations for mobile robots as that presented here and extract 2-D information from 3-D line segment data.

To recognize a room in a building, MARVEL uses the distribution of large objects about that room, as represented by matched stereo features, to characterize it. The important aspects of the data are the existence of the room features and their distribution about the robot. A 2-D representation of the stereo data obtained by projecting it onto the groundplane contains exactly this essential information. The 2-D data is further abstracted through the use of an occupancy grid representation. The simplicity of this method helps make the data robust with respect to partial occlusion and insensitive to the positional uncertainties of the stereo data, since the representation is not sensitive to these variations.

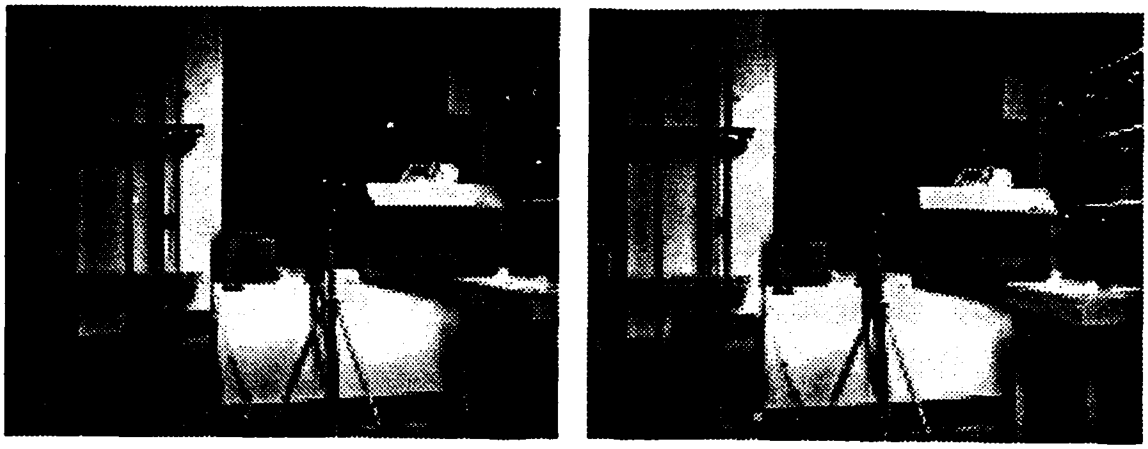


Figure 4-1: A stereo pair taken from Room 914 in the MIT Artificial Intelligence Laboratory.

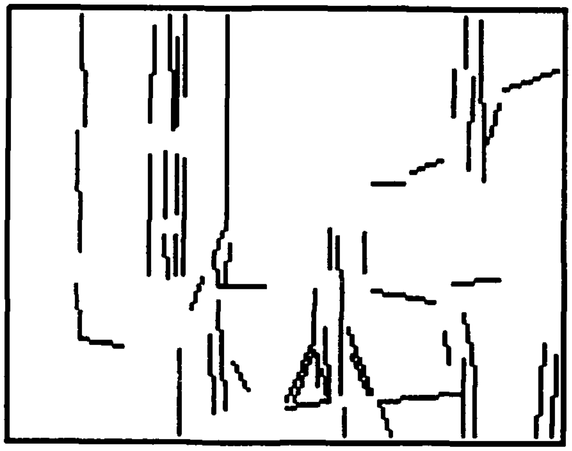


Figure 4-2: Matched stereo edges for the stereo pair in Figure 4-1. Each edge point is labeled with its 3-D coordinates.

4.2 Basis of the Model/Data Representation

As mentioned in Chapter 2, the input data for MARVEL is extracted from stereo pairs of images. For each stereo pair (e.g., Figure 4-1 taken from Room 914 in the MIT Artificial Intelligence Laboratory), a set of matched 3-D edges are found (Figure 4-2). The significant vertical edges are extracted (Figure 4-3) and their midpoints projected to the groundplane (Figure 4-4). This process is repeated for each rotation of the cameras until complete 360° coverage around the robot is obtained (Figure 4-5). These projected midpoints of 3-D vertical edges form the basis of the representation for the room.

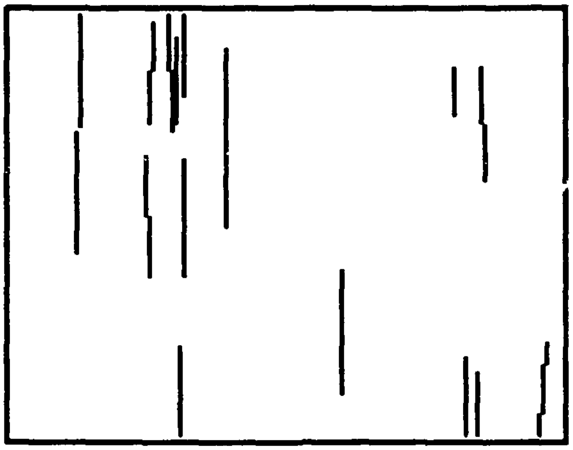


Figure 4-3: Significant vertical edges extracted from the edges in Figure 4-2.

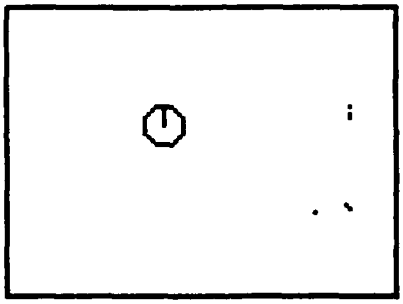


Figure 4-4: Projected midpoints of the vertical edges in Figure 4-3. The icon indicates the robot position and orientation.

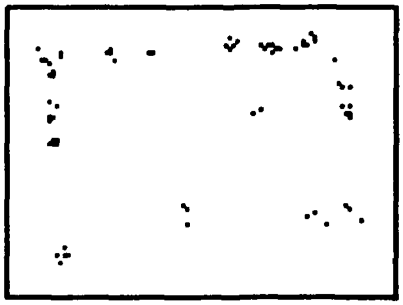


Figure 4-5: Projected feature midpoints for full 360° coverage around the robot.

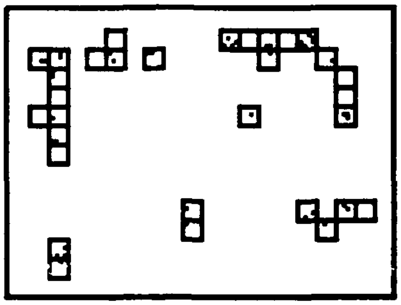


Figure 4-6: Occupancy grid squares that contain at least one projected vertical midpoint from Figure 4-5.

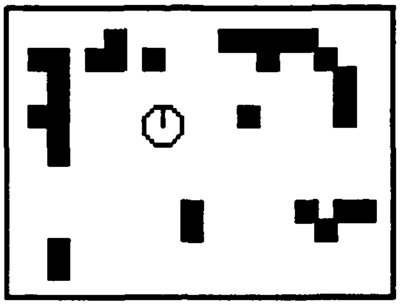


Figure 4-7: The set of data used to represent Room 914, which was derived from the projected midpoints of Figure 4-5.

4.3 Grid-Based Data Abstraction

The data obtained from the stereo input is further abstracted through the use of an occupancy-grid representation [Nilsson 1969] [Moravec and Elfes 1985]. A square grid is imposed on the groundplane over the projected vertical midpoints. Any grid square that contains a midpoint is then marked as occupied (Figure 4-6). (The fact that only midpoints of the vertical edges were projected guarantees that only one square will be marked per vertical.) The marked 2-D grid squares are used to represent the input data for the room (Figure 4-7 shows the data for Room 914). Each grid square is a data point with an associated 2-D position with respect to the robot-centered world coordinate system. Notice that the essential character of the input data—the distribution of features about the robot, has been retained in the grid-based data abstraction.

Each data point is labeled with a 2-D (x, y) position. The origin of the data coordinate system is the position of the robot when it acquired the data. The initial direction that the robot was facing defines the positive direction of the y-axis, with the positive x-axis extending to the right. The position of the robot in the room, then, is determined with respect to the stereo features that it finds.

Since MARVEL builds its models autonomously from its input data, the model representation will be very similar to the data representation. In fact, the models are represented by grid squares, as just described, that are weighted according to their importance to recognition. The model representation will be described more fully in Section 4.5.

Note that this grid-based representation does not look like the rooms that we see. Once again it is evident that the world is defined by the information our sensors give us and not by our concepts of the world. A world defined by stereo edges will not be the world populated by people, desks, and bookcases that we know. Through the grid representation. the world appears to MARVEL to be a collection of points with fixed positions distributed about the robot.

4.4 Advantages of a Grid-Based Representation

The grid-based representation for a room retains the essential characteristics of the sensor input that are needed for recognition. Specifically, the existence and distribution of significant stereo features are preserved. The 2-D (floorplane projection) positions of the features as determined by the stereo algorithm are represented by the positions of the data points (the grid squares).

The 2-D nature of the data representation reflects the fact that the mobile robot travels on the floorplane—its motion degrees of freedom are 2-D translations and rotation. The important task for the robot is room recognition and this task is simplified by the knowledge that arbitrary 3-D translations and rotations are impossible. Thus, a full 3-D recognition system would first limit the possible transform space to exactly the 2-D set of possibilities made explicit by MARVEL's 2-D representation. Two-dimensional data is also appropriate for MARVEL's subsidiary task of determining its position (on the floorplane) in the recognized room.

The compression to two dimensions of the three-dimensional stereo data reduces the complexity of the data. Faster processing algorithms can be designed to deal with 2-D data than with 3-D data due to this complexity reduction. The reduction of the recognition transformation space from six degrees of freedom to three has already been mentioned. Hardware speedups based on this reduced complexity are also possible (Chapter 12).

By compressing the stereo data to two dimensions, some insensitivity to occlusion is obtained. For example, assume that the 3-D edge of a doorway is in a room model. If a chair occludes part of this edge from a certain viewpoint in the room, then the recognition system must decide whether the visible portion of that edge corresponds to the full edge in the model. Since this decision would be based both on the length of the edge as well as its position, the visible part of the occluded edge might be small enough to be labeled a bad match for the model edge. In the 2-D representation, only the position of the edge is considered and the match is allowed. Obviously, features that are aligned vertically will be combined in the 2-D representation, but this poses no problem since only their 2-D positions are being considered.

The grid-based representation has advantages over using the exact 2-D positions of features when data uncertainty is considered. Since the location of a stereo edge has some uncertainty associated with it, a good deal of effort could be expended by the recognition system to get the best possible match of model to data. The grid squares provide a way of capturing this uncertainty in the representation itself. A one-foot grid size was used in the main part of the testing of MARVEL. (Results with other grid sizes will be discussed in Section [illegible].5.) Any displacement of a feature within this bound is ignored since the feature will still give rise to the same data square. A larger displacement would result in the instantiation of a neighboring data square. Since nearby data squares are considered during the matching process (Chapter 5), data resulting from stereo features with more uncertain positions will still be considered in the recognition process, albeit less strongly (in agreement with the lower certainty). The end effect of the grid abstraction, then, is to reduce the sensitivity of the recognition system to small perturbations of the positions of the data while maintaining sensitivity to the large displacements that occur between different features.

Since the stereo features are represented by grid squares, nearby features can be clustered into the same data square. This many-to-one data reduction benefits the recognition process since MARVEL does not need to devote processing to determining to which of several nearby model features a particular data feature corresponds. The fact that it corresponds to *one* of them suffices since recognition does not hinge on any one model or data point.

If two or more stereo features fall in the same grid square, that square is marked in the same way as if only one feature had fallen in it. No tally is kept of the number or sizes of features that give rise to a data point. (Take, for example, the case of a feature that appears in the overlap region shared by subsequent rotations of the stereo cameras. It might appear at two slightly different 3-D positions due to the finite stereo accuracy. The fact that these two features fall into the same grid square should not confer any additional importance on that square.) Equal weighting of the data points is in keeping with the premise of the recognition algorithm: it is the *distribution* of positions of the data that is important for recognition. Similarly, MARVEL does not assign an importance to a particular data square based on the features that caused it because the feature attributes are not a reliable indicator of their importance to recognition. On the other hand, the importance of individual model points to recognition can be determined and thus, the model points are assigned weights.

There is one situation, however, where it would make sense to vary the weights of the data grid squares. In Chapter 2, it was seen that the 3-D position provided by a stereo vision algorithm for a viewed object becomes less certain as the distance of that object from the viewer increases. When this uncertainty exceeds the size of a grid square, two or more grid squares could be marked to denote the location of the uncertain stereo feature. The weights and distribution of these marked squares could be determined according to the probability distribution of the uncertainty, which is characterized by the stereo geometry [Matthies and Shafer 1986]. The total weight of

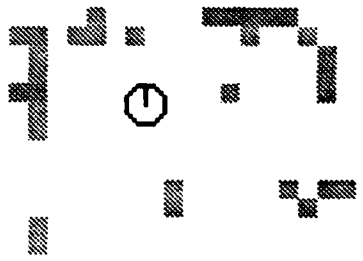


Figure 4-8: The initial model used to represent Room 914, derived from the data shown in Figure 4-7.

the data squares used to represent such an uncertain feature would be normalized to unity so that no special significance would be accorded to this group of points. This is consistent with the premise that importance should only be accorded to points that are important to recognition. This importance is represented by weights associated with individual model points.

4.5 Weighted Models from Grid-Based Data

As stated above, the models that MARVEL uses, which are built autonomously from its input data, must necessarily have a form similar to that data. The models are, in fact. built on a grid representation as is the data. In the models, however, the grid squares each have an associated weight. This weight reflects the importance of the model point to the recognition process. Since the robot operates autonomously, the determination of the importance of the model points must be done by MARVEL itself and not supplied by an outside agent.

The initial model for a room is simply the first set of data acquired from it and the coordinate system associated with the model is that of this first data set. The weight of each model point is initialized to unity to reflect the lack of knowledge about the point's importance to recognition of that room. The initial room model for Room 914, derived from the data of Figure 4-7, is shown in Figure 4-8. The model points are lightly shaded, indicating their low weights. In other models the more heavily weighted points will be indicated by darker shadings.

Through use of a model, it will become apparent that some of its points are important for recognizing the corresponding room. The model points are emphasized by increasing their weights. Likewise, some model points will be less important or even extraneous. The weights of these model points will be reduced or the model points will be eliminated entirely. This process of updating models based on successful recognition against them will be explained more fully in Chapter 7.

4.6 Summary

The key point of this chapter is that, for a robot, the world is defined by the information its sensors provide and not by our (human) concepts and understanding. Given the form of the input from the stereo system (3-D vertical line segments) and the task to be performed, a representation was described that makes explicit the essential information in the data that is needed for recognition. The representation chosen is an occupancy grid, which explicitly represents the 2-D positions of the stereo data. The grid representation provides insensitivity to small perturbations of the data positions while retaining the position information needed to distinguish separated data points. The models used for recognition are based on this data and the model representation is therefore similar to the data representation. The next chapter will explain how the grid-based data and weighted model are used to recognize a room in the robot's world.

Chapter 5

Recognition Against a Single Model

Given a grid-based model and set of data, how can it be determined if the data matches the model? Since the data and model are both assumed to be coplanar in the floorplane, a 2-D translation and rotation must be found that will best bring them into correspondence. This correspondence must then be evaluated to determine if the data indeed corresponds to the model. This chapter discusses the problem of matching a set of data to a single model and evaluating that match. Chapter 6 discusses recognition of a data set against a database of models.

The recognition process that MARVEL employs has three parts. First, a set of candidate alignments, each a 2-D transformation consisting of a translation, (x, y), and rotation, θ, is found for matching the data and model. The alignments are evaluated and the best candidate transformations are further refined using a least-squares minimization technique. The best transformation is then chosen and the model/data correspondence is evaluated under a set of recognition criteria. If the criteria are satisfied, the data is considered to be recognized as corresponding to the model. The accepted transformation then represents the displacement of the robot from the model coordinate system origin.

5.1 Initial Alignments

The transformation space for determining an alignment between a model and a set of data is three-dimensional (actually toroidal) with degrees of freedom in the x, y, and θ directions. Blind search through this transformation space to find the correct alignment, then, is obviously not the recognition scheme of choice. One could try every possible matching of model points to data points, but this still leads to an exponential number of possible matches. What is needed is a way to choose the model/data matches that are most likely to be correct so that further processing can be concentrated where the possibility of success is greatest.

Different approaches to the problem of determining initial alignments of model and data have been explored by [Ayache and Faugeras 1986], [Bolles and Cain 1982], and [Ullman 1986], to name a few. The approach used in MARVEL is to find larger-scale groupings of the individual model and data points and then to match these groupings to obtain the candidate model/data alignments. This approach is similar to that of [Lowe 1987] where "perceptual groupings" were used to select primitives that were likely to arise from the same object in a complex scene. The perceptual groupings were then used to obtain initial model/data alignments for later verification. In MARVEL, the groupings consist of linear clusters of data (or model) points.

5.1.1 Linear Data Clusters

Objects in indoor environments tend to lie in straight lines. This is an artifact of the man-made environment, since furniture, pictures, doorways, etc., tend to lie along walls. MARVEL exploits this situation by searching for linear clusters of data points. (For a consideration of other possible data point groupings, see Chapter 12.) Linear clusters of data and model points can be matched to provide initial alignment guesses for the recognition algorithm.

Given a set of data, MARVEL finds all possible linear groupings of data points. Obviously, any two data points define a line, but the linear groups found can be ranked by the number of points they include. The method used to find these linear groups is to find all pairs of points and enter the lines that they form into a hash table, indexed by their ρ and θ descriptors, where ρ is the perpendicular distance of the line from the origin and θ is the counterclockwise angle between the x-axis and the line. (For a discussion of the utility of the ρ-θ description of a line, see Chapter 3 of [Horn 1986].)

The hashing procedure uses a Hough bucketing table [Hough 1962] [Ballard and Brown 1982] indexed on ρ and θ. The size of each bucket is $\Delta\rho$ feet × $\Delta\theta$ feet, with the first bucket extending from $\rho = 0$ to $\rho = \Delta\rho$ and from $\theta = 0$ to $\theta = \Delta\theta$ All pairs of points are considered and the ρ-θ line defined by each acceptable pair is placed into its corresponding bucket. The Hough buckets (and thus the lines) are then ranked according to the number of point pairs that contributed to each bucket.

The bucket size used in the Hough clustering for MARVEL is 1.5 feet × 5°. For two data points to be acceptable for contributing a line to the Hough clustering, they had to be between 1.1 and 10 feet apart. The θ step size of 5° gives good coverage of the possible line orientations. The ρ step size of 1.5 feet avoids biasing the Hough clustering toward lines along the rows and columns of the data grid, which are spaced at 1 foot intervals. The 10 foot maximum distance between linearly grouped data points gives a rough locality measure so that accidental alignments of distant points are largely ignored, while the 1.1 foot minimum distance eliminates trivial lines from adjacent data points. Although all of the lines found by this clustering could be used in the determination of the alignments for a set of data, processing is focused on the most reliable alignments by considering only the ten most frequently occurring lines. Figure 5-1 shows this set of linear clusters for the Room 914 data shown previously in Figure 4-7.

The same procedure is used to find linear clusters of model points. Since the weights of the model points reflect their importance to the recognition process, however, the weights are also taken into account when finding the linear groupings. The "vote" added to a Hough bucket by a pair of model points is equal to the product of the weights of the model points. For example, if both model points in a pair has a weight of one, then the line formed by them only counts once. If one of the model

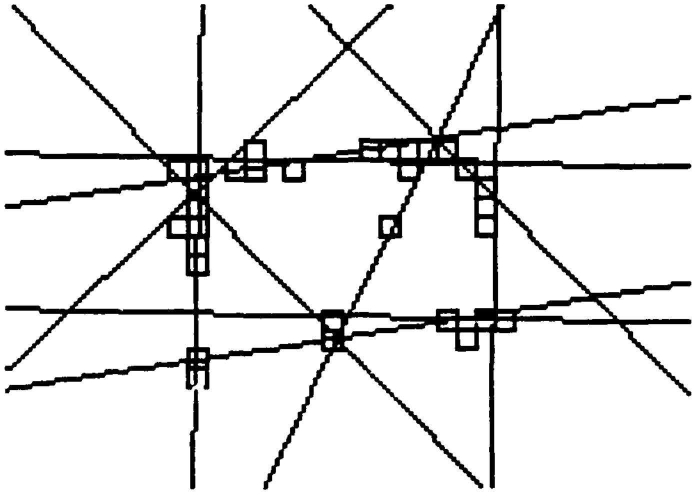


Figure 5-1: The ten most frequently occurring data point clusters for the Room 914 data previously in Figure 4-7.

points had had a weight of two, then the pair would have produced a vote of two, just as if the more heavily weighted model point had been two different model points at the same position. By taking the model point weights into account, the important model points more heavily influence the groupings (and thus the initial alignments) than the less important ones.

5.1.2 Initial Alignments from Linear Cluster Pairs

If a data linear cluster is aligned with a model linear cluster, there is still one degree of freedom for the model-data alignment: the model and data can "slide" over each other along the line. A second matching of a model and data line, however, can eliminate this freedom and fix the alignment of model and data. Since an intersecting pair of lines defines a unique point and direction, the initial alignments of model and data are based on matches of pairs of model lines with pairs of data lines.

Each non-parallel pair of linear clusters found for a model or data set determines a unique point: the intersection point of the lines. When a pair of model lines is matched with a pair of data lines, these intersection points are matched and the model rotated about this point to make the model and data lines colinear. If the lines of a pair are almost colinear, however, this intersection point is very sensitive to small perturbations of the lines. Thus, all possible pairs of the most frequently occurring lines are found and ranked in decreasing order according to the angle they form. (Pairs forming less than a 20° angle are eliminated outright.) The top ten line pairs from the model and from the data are then compared to determine the initial alignments. Each pair defines a unique point, the intersection point, and direction, the bisector of the two lines. Actually, each line pair has two bisectors defining four directions. The bisector ray that falls in each "quadrant" formed by the line pair is a possible direction to be associated with the pair. The direction for the pair is chosen as the bisector ray that falls in the same quadrant as the center of mass of all the data (weighted model) points. If the center of mass is too near the intersection point, then each of the four possible direction/intersection point pairs are used.

Given a model line pair and a data line pair, the 2-D translation and rotation to bring the pairs into alignment are found. (The angles between these pairs of lines may not match exactly, but if they agree within 10 degrees then the match is accepted.) First, the rotation of the model that will make the bisectors parallel is determined. Since the bisectors are rays, this rotation can be uniquely determined so that the rays point in the same direction. Then, for this rotation a translation of the model is found that brings the intersection points into coincidence. Each such 2-D rotation and translation determines an initial alignment for the model and data. Each alignment can be quickly checked for feasibility: if the model and data points are sufficiently elongated (eccentricity of an ellipse fit to the points greater than .7), then an alignment must bring the major axes of the points within 45° to be a feasible initial alignment. The feasible alignments are ranked based on how well the model and data angles agree. During testing, the best alignment usually occurred as one of the first few in this ranking, so only the top ten are kept for further refinement using a least-squares minimization technique (Section 5.2).

After obtaining a new set of data for Room 914 from a different position and orientation in the room (Figure 5-2), a set of initial alignments was found for this data against the initial Room 914 model (Figure 4-8). The alignment shown in Figure 5-3 led to correct recognition for this data and model. Figure 5-4 shows an

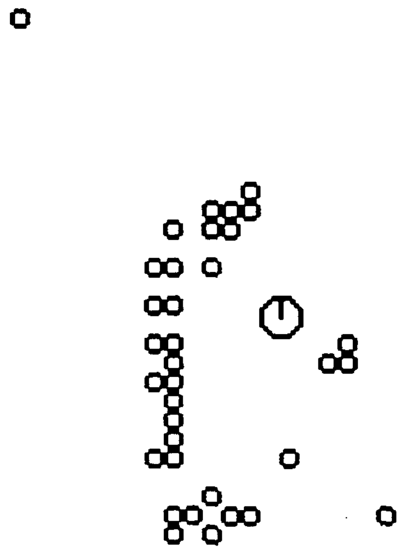


Figure 5-2: A new data set taken from a different position and orientation in Room 914.

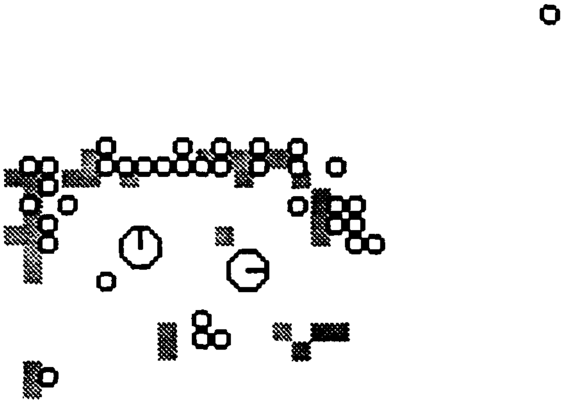


Figure 5-3: The initial alignment of the data from Figure 5-2 with the initial Room 914 model (Figure 4-8) that resulted in correct recognition.

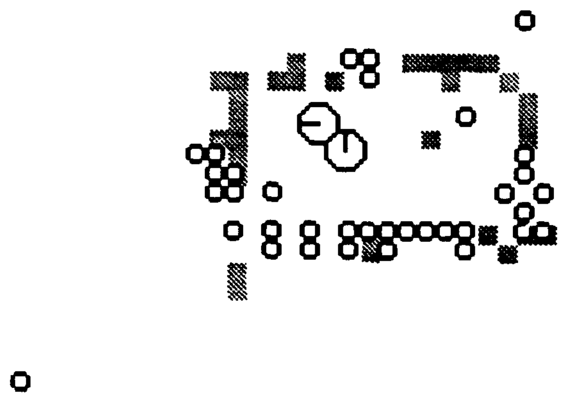


Figure 5-4: An incorrect initial alignment that was found for matching the data from Figure 5-2 with the initial Room 914 model (Figure 4-8). It was eliminated in later stages of the recognition process.

example of an initial alignment that was later eliminated because it did not lead to correct recognition.

5.2 Least-Squares Transform Refinement

The model/data alignments obtained from the linear clusters are only rough guesses of the correct transform to match model to data. Each alignment must be refined for maximum model/data coincidence (see below) and then evaluated for goodness of match (see Section 5.3).

Given a transform estimate from an initial alignment, the transform refinement procedure is

1. Improve the initial alignment as much as possible at the current search radius:

    (a) Given an initial alignment transform, transform each model point.

    (b) Find the closest data point within the current search radius for each transformed model point (if such a data point exists).

    (c) Use the set of corresponding data and transformed model points to improve the transform estimate using a least-squares minimization.

    (d) Evaluate the improved transform. If the evaluation criteria are satisfied, continue, otherwise, loop back to Step 1a for further transform improvement.

2. If this is the final search radius, stop. Otherwise, reduce the search radius and loop back to Step 1.

The procedure outlined above for the least-squares transform refinement is given in more detail in Appendix A.

The search radius set used by MARVEL is (3, 2, 1). The initial large search radius allows matching model and data points to be found for rough initial alignments. As the alignments improve, the search radius is reduced to eliminate extraneous matches and permit a better estimate of the model/data transform. As a quick check for the goodness of an initial alignment, if an alignment does not match at least half of the weighted model points at the largest search radius, then that initial alignment is eliminated from further consideration. This check is based on the sum of the weights of the matched model points instead of a simple count.

The least-squares minimization method used to refine the transform estimates assumes that the parameters being adjusted are linear. The x- and y-translations are indeed linear, but this is not the case with the rotation. Instead, the minimization is done about an operating point (x, y, θ), which is the initial transform estimate. For small changes in θ about this point, $\cos\theta$ and $\sin\theta$ are almost linear. Because they are not exactly linear, however, the result from the least-squares minimization does not actually correspond to the minimum error possible between the data points and the transformed model points. Thus, we repeat the minimization (Steps 1a through 1d) using the newly derived transform as the new operating point. This process continues until the match between data and transformed model points is good enough.

The criteria used in Step 1d to determine if sufficient improvement in the data/transformed model match has occurred are based on the weighted transform variance. The weighted transform variance is the weighted sample variance S^2 as determined by summing the weighted squared distances between each model point and its closest data point, then dividing by the number of model points minus one [Walpole and

Myers 1978].

$$S^2 = \frac{\sum_{i=1}^{n} w_i(\mathbf{d}_i - \hat{\mathbf{m}}_i)^2}{n-1}$$

where

$$\begin{aligned} \hat{\mathbf{m}}_i &= \text{the transformed model point } \mathbf{m}_i \\ \mathbf{d}_i &= \text{the data point that } \hat{\mathbf{m}}_i \text{ matched} \\ w_i &= \text{the weight assigned to model point } \mathbf{m}_i \end{aligned}$$

The weighted transform variance is checked after each iteration through the least-squares minimization. If either the variance becomes small enough ($S^2 < \delta_1$) or changes slowly enough $\Delta S^2 < \delta_2$, the iteration process is stopped and the last transform determined is accepted as the best transform possible for this initial alignment at the current search radius.

The least-squares transform refinement described above is performed on each initial alignment for the given model/data pair. The final transforms are then evaluated to determine which yields the best fit between the model and data points. The best transform is taken to be the one that best accounts for the model: the weights of the matched model points are summed and the transform that produces the highest weighted sum is chosen. If two or more transforms produce the same weighted sum of model points, the one with the smallest transform variance S^2 is chosen as the best fit.

Note that in the transform refinement procedure, all possible matches are found using the current transform estimate before a transform update is calculated. Some researchers have been using an incremental approach to refining an initial transform (e.g., [Lowe 1987], [Ayache and Faugeras 1986]). This approach was tried for MARVEL with disastrous results. For a given initial alignment as found above, the alignment should be most exact at the intersection points of the line groups (the base point of the alignment) and get progressively worse farther from this point. Therefore, for the incremental approach it would be reasonable first to refine the transform estimate with model/data matches close to this point and then proceed outward as

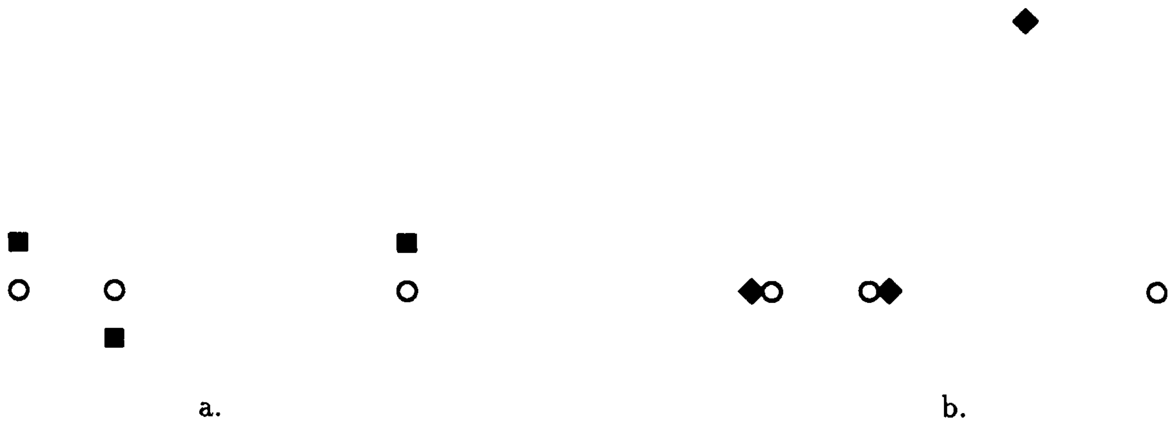


Figure 5-5: An example of how an incremental transform update can take a good initial transform estimate (a) and render it completely incorrect (b). Model points are shown as squares and data points as circles.

the transform estimate becomes better. Consider the situation shown in Figure 5-5, however, where the squares are model points, the circles are data points, and the base point of the alignment is at the left. Figure 5-5.a shows the model/data alignment immediately after the initial transform estimate has been applied to the model points. (For clarity, the remaining model and data points are not shown.) If the transform is incrementally improved by first using the left and middle point matches, and this improved transform applied to the model points, the left two model points are shifted slightly to become better matched with their corresponding data points, while the rightmost model point is drastically rotated away from its corresponding data point (Figure 5-5.b). The rightmost model point now cannot be matched with the rightmost data point because of the increased distance between them. What was a near-perfect initial transform estimate has now become a completely incorrect updated transform. Using both an incremental least-squares transform update and a Kalman filter approach (which is inherently incremental), just this sort of transform deterioration was observed during the testing of MARVEL. For this reason, all possible model/data point matches are found before updating the transform estimate.

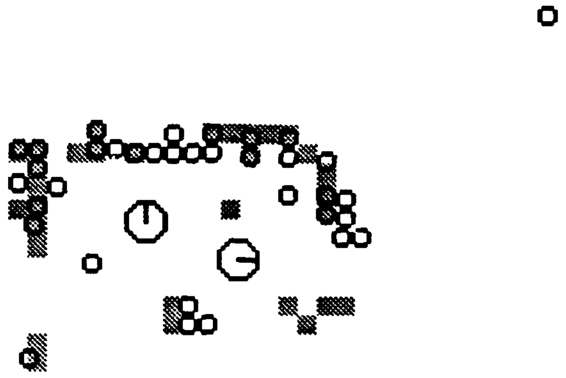


Figure 5-6: Recognition of Room 914 data set shown in Figure 5-2 with the initial model of Room 914 from Figure 4-8.

5.3 Transform Test

After the least-squares transform refinement has been completed for each initial alignment of model to data, the transform that best matches the model to the data must be chosen from the candidates. That best match is chosen as the one that accounts for the largest part of the model: at the smallest search radius, all model point matches are found and the weights of the matched model points are summed. The transform with the highest sum of matched model point weights is chosen as the best match out of the candidates. This is in keeping with the concept that the model point weights reflect the importance of individual model points to recognition. In the event of a tie, the transform that matches the most data points is chosen (see Figure 5-6). Note that since the data coordinate system includes the current position $(0,0)$ and orientation (pointing along the positive y-axis) of the robot, this transformation also represents the position of the robot with respect to the model coordinate system.

The transform found represents the best possible match of the data to the currently selected model. It still remains to be determined, however, if this is the *correct* model. Chapter 6 explains how the matches of several models to the data are compared to choose the correct match, if one exists. (Recall that since one task of a mobile robot is exploration, the correct model may not yet be in the database.)

5.4 Summary

This chapter discussed the problem of matching a set of data to a single model and evaluating that match. To accomplish this match, initial model/data alignments were found, based on groupings of the data and weighted model points. The transform for each initial alignment was then refined based on individual points using a least-squares minimization technique. The best possible match of model to data was then chosen from this set of candidate transforms. The next chapter will explain how the correct match (if it exists) is chosen from possible matches to a set of models.

Chapter 6

Recognition Against a Database of Models

The procedure for matching a set of data against a single model was discussed in Chapter 5. This matching process determines the 2-D transform that best fits the model to the data. It does not, however, determine if the model/data match is *correct*. To achieve recognition, MARVEL must compare the current data to a set of candidate models and then determine which, if any, of the current matches are correct.

As the robot explores its world, MARVEL builds up a database of known locations. The current location could be one of these new locations or it could be a location not yet encountered. The recognition process must make this decision. By taking advantage of the fact that MARVEL operates in conjunction with a mobile robot, however, the full database of models need not be searched. Since the robot is following a world model (see Chapter 9), its current location estimate constrains the database search. The candidate models for recognition are limited to the model for the current (estimated) location and the models for the nearby locations.

Using this limited set of candidate models, MARVEL must decide if the current location corresponds to a known location. Since the current location might be new to MARVEL, however, the best-fit model cannot simply be chosen. Instead, independent model evaluations are made and if no model passes the recognition test, then the location is declared to be a new one and a model for it is built from the current data and added to the model database.

6.1 Recognition Test

Unlike many recognition systems, MARVEL cannot depend on having complete and accurate models. Since the models are built autonomously from input data, they will often be missing some location features or contain errors that originated in the stereo system. Also, the models can never completely reflect the important features in the locations since we allow the locations to change. Thus, to determine how well a particular model matches the current data, the recognition test considers both how much of the data is matched by the model and how much of the model is matched by the data.

To determine how well the data is matched by a model, MARVEL counts the number of data points that have a matching model point. This matching process is performed for the smallest search radius of the least-squares transform refinement described in Section 5.2 ($r = 1$). The percentage of the data points matched by the model is the first recognition criterion.

The number of model points with a matching data point is used to determine how well the model is matched by the data. Since some model points are more important than others to recognition, however, the *weights* of the matched model points are summed to judge the quality of the match. This weighted sum was calculated during the transform test performed on the model/data match (Section 5.3), again at search radius $r = 1$. The weighted sum of the matched model points as a fraction of the total weight of the model is used as the second matching criterion.

To evaluate a model/data match, the percent data matched and the percent weighted model matched are considered. Each model is evaluated independently. For a given model, if at least 80% of the data and 50% of the weighted model are matched, then that model is recognized and the robot is in the location corresponding to that model. (The selection of these thresholds is discussed in Section 8.2.) If no model passes these threshold tests, then the robot believes it is in a location it has

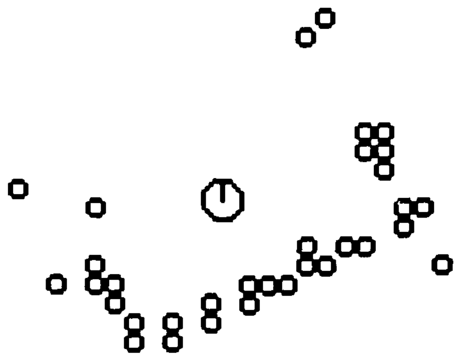


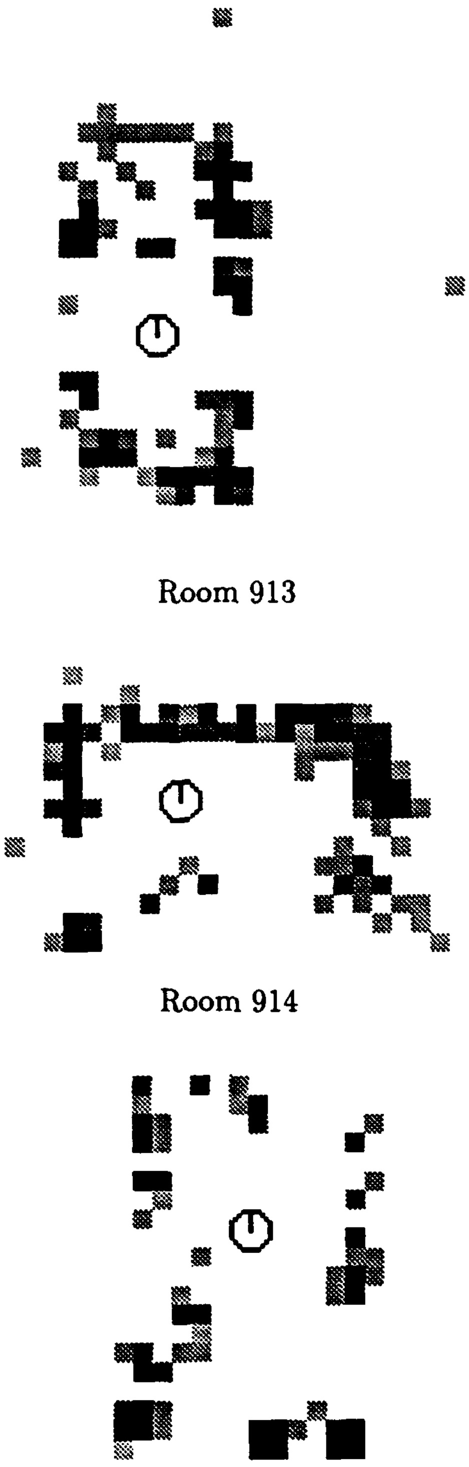
Figure 6-1: A data set to be recognized.

never encountered before and the current set of data is used to create a new model that is entered into the database. In the unlikely event that two models pass the threshold test (this never occurred during testing), the robot is moved to a different position in the room and a second set of data obtained. This second set of data is combined with the first and recognition performed with the combined data against the competing models. The improved data set should remove any ambiguities. This same method is also used if no model passes the recognition test, but one or more are very close to passing the thresholds (see Section 10.2 for more details).

The data and weighted model recognition thresholds were determined heuristically by considering recognition results between the available models and data. The prime consideration in choosing the thresholds was to avoid any false positive recognition occurrences. After these minimum thresholds were determined, the recognition thresholds were chosen to keep the false negative rate acceptably low (see Section 8.2) while staying safely away from the break over into false positive recognition.

6.2 Example

As an example of recognition against the model database, consider the data shown in Figure 6-1. The estimated position of the robot is Room 914. Thus, the models for Room 914 and the nearby locations, Rooms 913 and 915, are chosen as candidates for recognition from the database (Figure 6-2). The best match possible is obtained between the data and each of the candidate models (Figure 6-3). Each model/data

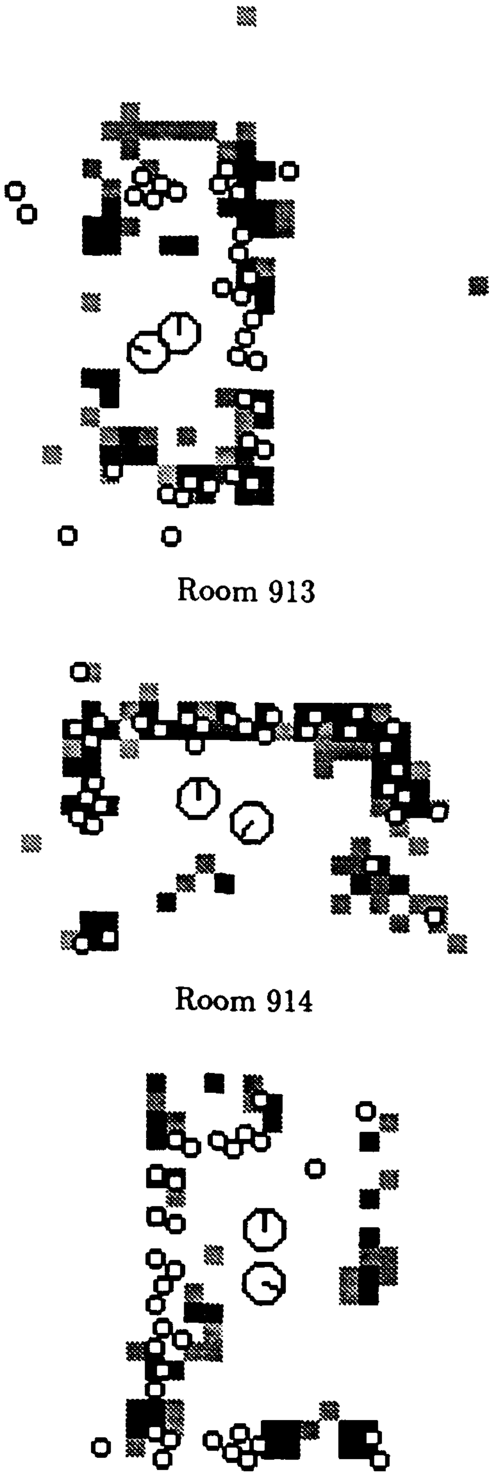


Room 913

Room 914

Room 915

Figure 6-2: Candidate models from the database for recognition of the data set shown in Figure 6-1.



Room 913

Room 914

Room 915

Figure 6-3: Recognition against the candidate models shown in Figure 6-2.

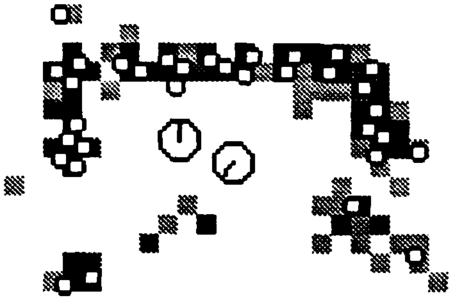


Figure 6-4: Correct recognition chosen from candidates in Figure 6-3.

match is evaluated independently to determine which, if any, succeeds. The match for Room 914 passes the recognition criteria, so the data is recognized as coming from this room, i.e., the robot's location is determined to be Room 914 (Figure 6-4). A brief inspection of the match of the data to the Room 914 model should satisfy the reader that MARVEL made the correct decision. During testing, this set of data was indeed taken in Room 914.

6.3 Summary

This chapter described the procedure for recognizing the current location with respect to a database of models. The choice of candidate models is constrained by the estimate of the current location of the robot. Independent evaluations are made on this restricted set of models to determine which, if any, match the current set of data. If no match passes the recognition criteria, then the robot is assumed to be in a newly-encountered location and a model for it is built from the data. If recognition has succeeded against an existing model, then the current data is used along with the recognition transform to update that model. This model update process is described in the next chapter.

Chapter 7

Model Update

Given successful recognition. the information from the current set of data is used with the recognition transform to update the recognized model. This model maintenance serves to reinforce the important model features, respond to changes in the location represented by the model, and remove errors in the model. Thus, autonomous operation of a mobile robot is made possible by MARVEL's model update process.

7.1 Why Update the Models?

The models used for recognition are built autonomously by MARVEL. The ideal constituents of a model are just those features that are important to recognition. Since no *a priori* knowledge of the world is available to MARVEL, however, the important features cannot be immediately selected from the ones presented by the input system. Instead, the importance of the various model features is determined by their use. The more often a model feature is used for recognition, the more important it is to recognition and the more it should be relied upon.

One of the basic assumptions in this thesis is that MARVEL would operate in a world subject to change. If the appearance of a location changes, then the model used to represent that location must also change to enable recognition to succeed. If the world changes slowly enough, then there will be a sufficient number of features in common between the existing model and the current location, as represented by the input data, to enable recognition. The current data can then be used to update the model to be more consistent with the location as it now exists.

The exact amount of change allowed in a location between recognition attempts depends on the recognition parameters. The change must be small enough so that the total mismatch caused by the change and any input error does not cause the recognition test to fail. If the effect of the input errors is lessened through the use of two or more data sets (Section 10.2), then the amount of change allowed is essentially quantified by the recognition thresholds. Theoretically, for a 50% model/80% data recognition threshold (Section 8.2), a change of up to 20% in the perceived location features could be tolerated. In practice, the allowable amount of change is affected by the completeness of the data set, the existence of errors in the model and data, and the importance of the model features involved. Changes in location features that correspond to important model features affect recognition more than changes to relatively insignificant location features.

Occasionally, the stereo input provides data that, although valid, does not belong in a location model because of its ephemeral nature. Such data typically corresponds to non-repeatable effects such as shadow edges or people in the scene. The features that correspond to these ephemeral events will be incorporated into the model since no higher-level filtering process exists to recognize and eliminate them. The very reason that they are not desirable, however, makes them susceptible to removal by the model update process. Because these features only appear briefly over the life of the robot, they occur in only one or very few data sets. Since the features are not reinforced through repeated observation, they do not attain importance in the models and can be eliminated.

The stereo input system, like any sensory input system, will occasionally yield erroneous data. As with the ephemeral event data, these errors will be incorporated into the models. The temporary nature of these errors allow them also to be acknowledged as unimportant and removed from the models.

7.2 Overview of the Model Update Process

From the discussion above, it is apparent that the model update process should reinforce the importance of model points that are used frequently for recognition. Without being overly sensitive to occasional occluded data, this process should also remove model features that do not correspond to current location features. Obviously, these changes to the models must be based on the currently available data as well as the existing information contained in the model.

To yield robust recognition, a model must well characterize the location it represents. In terms of recognition, the model features that best characterize a location are those features that are important to achieving successful recognition. This importance is determined autonomously by keeping track of how often the model points are used in recognition. MARVEL accomplishes this by assigning weights to the model features based on their frequency of use. These feature weights are incorporated naturally into the least-squares transform refinement of the recognition process (Appendix A).

Since the importance of the model points is determined through their use in recognition, the model update process should use the current data and recognition result to update the matched model. The weights of model points that are repeatedly used for recognition should be increased, while weights of rarely used points should be decreased. The logical extension of this process is that points that are no longer used for recognition should be removed from a model entirely. Also, if a new feature is seen in the data that does not already exist in the model, the update process should add it to the model, albeit with a low confidence (weight) since the feature has only been seen once.

Even though the model should represent the importance of its constituent points to recognition, the importance of any one point must be limited. Without such a maximum weight, a point could become so entrenched in a model that it could essentially never be removed even if the current data repeatedly confirms that its corresponding location feature no longer exists. This limit on importance is reflected

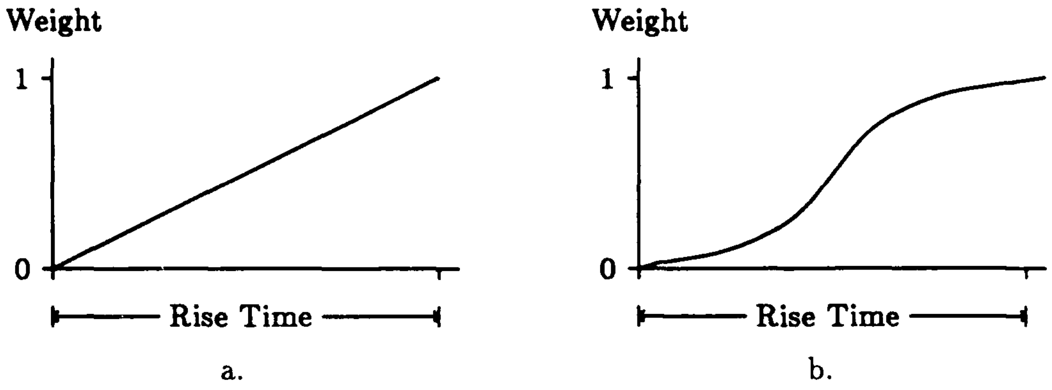


Figure 7-1: Functions used to update the weights of model points. a. Linear b. Exponential

by allowing the model point weights to vary between 0 and 1.

The rate at which model point weights are updated is determined by the *weight update function*. The weight update is based upon the number of times that the point was seen. The count c for a model point is increased by one each time the point is seen in the data and decreased by one each time it is not. A linear update function was used for the main part of the testing of MARVEL (Figure 7-1.a). Given a model $M_{j-1} = \{m_{i,j-1}\}$, which incorporates the information from data sets $D_1, D_2, \ldots, D_{j-1}$, the weight assigned to the i^{th} model point $m_{i,j}$ in model M_j, based on the information from data set D_j, is

$$W_{m_{i,j}}(c\,;t_r) = \begin{cases} \dfrac{c}{t_r} & 0 \le c < t_r \\ 1 & t_r \le c \end{cases} \tag{7.1}$$

where t_r is the rise time (the number of times a point must be seen to achieve full weight) and c is the number of times the point has been seen in data sets $D_1, D_2, \ldots, D_j$. With this update, a model point's weight is increased linearly each time it is used and decreased linearly each time it is not. The weight of a point is constrained to a maximum value of 1 and a point whose weight decreases to 0 is removed from the model entirely. The response of the update function is changed by varying the rise time.

The second update function tested had an exponential rise initially, described by a function of the form $e^{\frac{c}{t_c}}$. The second half of the function approaches the maximum weight of 1 at an inverse exponential rate, with the general form $1 - e^{-\frac{c}{t_c}}$ (Figure 7-1.b). Functions such as the right-hand part of this weighting function are typically described in terms of their time constant t_c: after one time constant, the function has attained 63% of its final value. The rise time of such a function is usually taken as three time constants, which brings the function to 95% of its final value. For the weight update function, this time constant must be halved since the right-hand part of the function only accounts for half of the model point weight rise time t_r. Likewise, the constants associated with these functions must also be adjusted in order to join the two exponentials smoothly. The resulting weight update function is

$$W_{m_{i,j}}(c\,;t_r) = \begin{cases} 0 & c = 0 \\ .5e^{\frac{c-t_r/2}{t_r/6}} & 0 < c < \frac{t_r}{2} \\ 1 - .5e^{-\frac{c-t_r/2}{t_r/6}} & \frac{t_r}{2} \le c < t_r \\ 1 & t_r \le c \end{cases}, \qquad (7.2)$$

where the values at $c = 0$ and $c = t_r$ have been forced to 0 and 1 (instead of letting these values be approached asymptotically). With the exponential update, the weight of a model point is increased gradually at first under the assumption that points seen only a few times may be erroneous. The importance of a fully-weighted point is decreased slowly at first, reflecting a reluctance to discard a model point that has been useful in the past.

In the same way that new location features must be added to the models, features that no longer exist in the locations must be retracted from the models. Points that are believed to be reliable, however, should not be removed quickly from a model. Since the reliability of a point is represented by its weight, model points are retracted by decreasing their weights until they simply "fade away." Model point weights are decreased according to the weight update function by reducing the count c for a point each time it is not seen. When the count (and weight) reaches zero, the point is removed entirely from the model.

Unfortunately, the absence of a location feature cannot be determined immediately from the input data. A missing feature in the input might have been obscured by some occlusion in the scene or lost due to an input error. Thus, the lowering of a model point's count should not happen immediately if it is not in the current data. Instead, a hysteresis is imposed so that a point's count c is incremented by one each time the point is seen, but decremented by one only if the point was last seen more than h data sets ago. This is accomplished by keeping an age a for each model point, which indicates how many data sets ago the point was last seen. An age of zero indicates that a point was seen in the current data set. Thus, a hysteresis of $h = 1$ indicates that a model point must have been seen in the current data set or in the immediately preceding data set to not have its count and weight reduced. If $h = 0$ and $t_r = 1$, the update replaces the model with the current set of data.

With each new data set. the information associated with each model point must be updated. First, the age of the model point is either reset or incremented depending on whether it was seen in the new data set:

$$a_{m_{i,j}} = \begin{cases} 0 & \text{if } m_{i,j-1} \in D_j \\ a_{m_{i,j-1}} + 1 & \text{otherwise} \end{cases} . \tag{7.3}$$

The count of the number of times that the point was seen is then updated, taken the age hysteresis into account:

$$c_{m_{i,j}} = \begin{cases} \min(t_r, c_{m_{i,j-1}} + 1) & \text{if } m_{i,j-1} \in D_j \\ c_{m_{i,j-1}} & \text{if } m_{i,j-1} \notin D_j \text{ and } a_{m_{i,j}} < h \\ c_{m_{i,j-1}} - 1 & \text{if } m_{i,j-1} \notin D_j \text{ and } a_{m_{i,j}} \geq h \end{cases} . \tag{7.4}$$

(Any point whose count goes to zero is eliminated from the model.) Finally, the weight $W_{m_{i,j}}(c_{m_{i,j}}; t_r)$ is determined by Equation 7.1 or 7.2.

The two model update parameters t_r and h affect how the models change over time. Larger rise times allow stable room features to play a bigger role in the recognition process. Smaller rise times downplay the importance of existing model points in favor of the incoming location data. A large hysteresis provides a greater insensitivity to data drop-outs due to occlusion, while a smaller hysteresis allow quick response to changes in the world. (These two parameters also interact to affect the persistence of model points that should be retracted.) Thus, the rise time and hysteresis should be chosen based on assumptions about the stability of the world and the reliability of the input data.

7.3 The Model Update Algorithm

The first step in the model update algorithm is provided by the recognition module: the model that corresponds to the data must be found and the transform that best aligns them determined. The data can then be used to update the model. The model update algorithm in described in this section, while the details of the algorithm are given in Appendix B. The linear weight update function was used to generate the models shown below, with update parameters $t_r = 4$ and $h = 4$.

7.3.1 Model Initialization

An initial model simply consists of the data points first collected for a particular room. The weight of a model point reflects its importance to recognition, but none of the initial points is known to be more important than any other since no recognition has been performed using them. Thus, each initial model point weight is initialized to $W_{m_{i,1}}(1; t_r)$ since the point has only been seen once ($c_{m_{i,1}} = 1$).

The initial model for Room 914 is shown in Figure 7-2 and is derived from the data points shown in Figure 4-7. The model points are lightly shaded, indicating their low weights. In other models the more heavily weighted points will be indicated by darker shadings.

7.3.2 Model Update

The model update algorithm uses the information from a new set of data and its recognition with a model to refine the model to reflect better the state of the location it represents. Since a new set of data for the location is in hand, the first step in the update algorithm is to increase the counts of the existing model points by one.

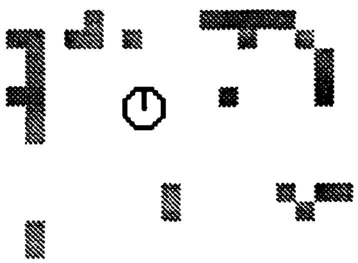


Figure 7-2: The initial model used to represent Room 914.

To combine a new data set D_j with the model M_{j-1}, the model/data point correspondences must be determined. Recognition provides a 2-D transformation (x, y, θ) that brings the model and data into correspondence. Using this transformation, the closest data point to each model point is found. (For testing, the maximum allowable distance δ_{min} was set to the smallest search radius of the least-squares transform refinement $r = 1$ (see Section 5.2).) The age of each model point is updated (Equation 7.3) based on whether it was matched in the new data set. A model point is matched if its corresponding data point is closer than δ_{min}:

$$m_{i,j-1} \in D_j \qquad \text{if } \exists d_{k,j} \in D_j \text{ such that } \|m_{i,j-1} - d_{k,j}\| < \delta_{min}.$$

The count and weight of each model point $m_{i,j}$ is then updated according to Equation 7.4 and Equation 7.1 or 7.2. Any points with a current count of zero are removed from the model.

Finally, new data is added to the model. Any data point that was not matched to a model point (within the maximum distance δ_{min}) is added to the model with weight $W_{m_{i,j}}(1; t_r)$ and age $a_{m_{i,j}} = 0$. All of the current data has now been incorporated into the model. Each model point has been updated by adjusting its age, count, and weight appropriately or by eliminating it from the model.

Figure 7-3 shows recognition of the initial model for Room 914 (Figure 7-2), with a new set of data. The recognition transform and new data provide the information

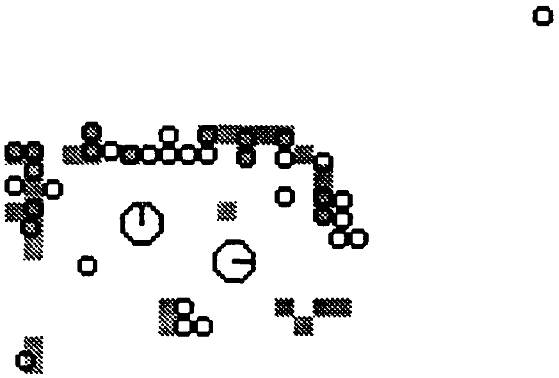


Figure 7-3: Recognition of the second set of data with the original model for Room 914 (Figure 7-2).

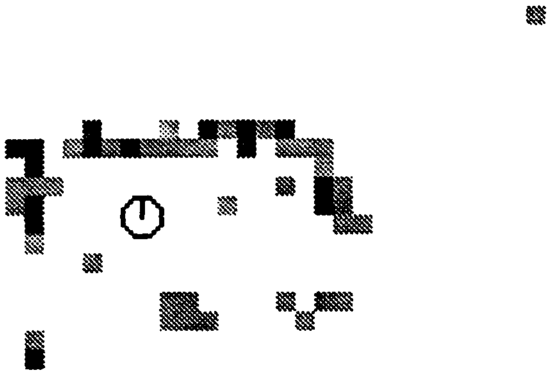


Figure 7-4: Room 914 Model 2, which is the updated initial model based on the recognition result shown in Figure 7-3.

necessary to update the model. Figure 7-4 shows the result of the first update to the initial model for Room 914. Comparing this new model with the original, the added and updated model points are obvious. No points have been retracted at this early stage in the model evolution. Recognition of a set of data with Room 914 Model 6 is shown in Figure 7-5 and the resulting updated model in Figure 7-6. In this model update, fading and deletion of old model points can be seen as well as reinforcement of existing points. Appendices C, D, and E show the data sets and model updates for Rooms 913, 914, and 915, respectively.

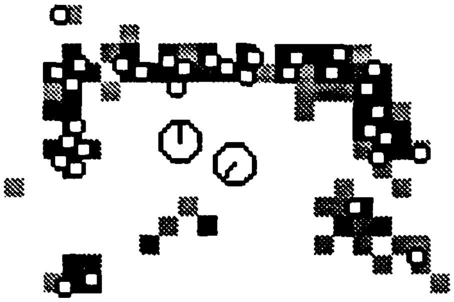


Figure 7-5: Recognition of the seventh set of data with the Room 914 Model 6.

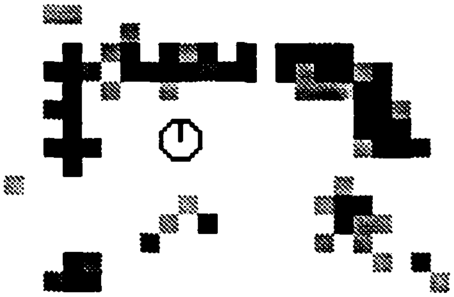


Figure 7-6: Room 914 Model 7, which is the updated Model 6 based on the recognition result shown in Figure 7-5.

7.4 Summary

Given recognition of a current set of data with a model that was built autonomously, the data should be used to update that model. The update enables the model to reflect the current state of its represented location as that location changes. Errors are also removed from the models by the update process. Since the update is based on recognition data, it should emphasize those model points that were used in recognition and are therefore important for recognizing the model. These goals are accomplished by the model update algorithm that was described above. It changes the weights of model points based on their use in recognition so that the recognition algorithm can be based on the relative importance of the points. The implemented update algorithm allows parts of the models to be added and deleted to respond to changing location features and to deal with the presence of error. By keeping track of how current the data is that produced the various model points, deletion of old or erroneous model points can be accomplished without undue sensitivity to occlusions in the input data. By keeping the models current with the state of the locations and relatively free from error, the model update process yields recognition that is more robust than is possible with static models. Results of testing MARVEL's recognition system using these evolving, updated models is discussed in the next chapter.

Chapter 8

Recognition Results

MARVEL was tested on real images taken on the ninth floor of the MIT Artificial Intelligence Laboratory using a stereo system (described in Section 2.3) mounted on a movable tripod. The results of this testing demonstrate that MARVEL successfully supports mobile robot navigation through recognition of world locations. This chapter describes the test world, explains ιhe testing procedure, and presents the results of the testing.

8.1 Test Procedure

MARVEL was tested on the ninth floor of the MIT Artificial Intelligence Laboratory using the stereo camera setup described in Section 2.3. The locations used for recognition were Rooms 913, 914, 915, and the ninth floor playroom, which will be referred to as Room 9PR. Rooms 913, 914, and 915 were chosen specifically as worst-case locations to challenge the recognition system. The rooms have almost identical dimensions (17′8″ × 10′5″, 17′8″ × 10′10″, and 17′8″ × 10′11″, respectively). This similarity forces the system to rely on the features found in the rooms and not simply on the different sizes. Room 9PR was added to the test set of rooms to demonstrate that MARVEL also performs well when comparing rooms of different sizes. (Room 9PR is 32′6″ × 22′3″.)

For each of Rooms 913, 914, and 915, twelve sets of data were used. One set of data was also used for Room 9PR. To build these data sets, a total of 1296 images (648 stereo pairs) were taken. For each room, Model 1 was built from the first set of data taken in that room (Section 4.5), with the position of the robot chosen arbitrarily. For each of these models, recognition was performed against every set of data except the one that was used to create the model itself. This first round of testing therefore comprised 144 recognition trials. For the second round of testing, Model 2 for each of Rooms 913, 914, and 915 was built from the first two data sets (Chapter 7). Recognition was then performed for each of these models against all data sets that were not used to build the model being tested. This second round of testing comprised 105 trials. This process was continued through Model 11 for each of Rooms 913, 914, and 915. (For each room, Model 11 was built from the first 11 data sets for that room, leaving one data set for that room not incorporated into the model.) All told, there were 1059 recognition trials of the recognition system. The results presented below are based on these trials.

8.2 Recognition Performance

When considering the recognition trials, there are two important questions to ask. First, how often does incorrect recognition occur? Second, how often is correct recognition possible, but missed? Incorrect recognition is a false positive. If false positive recognition occurs while a mobile robot is using MARVEL, the robot will think that it is in the recognized location instead of where it actually is. Since the purpose of MARVEL is to enable a robot to determine its location, this sort of error should be avoided. Since nothing comes for free, there will obviously be a tradeoff involved in keeping the number of false positives low. That tradeoff is an increase in the number of false negatives. This is reasonable, since if MARVEL is to be conservative in declaring a match between model and data, it will occasionally fail to achieve recognition when the correct model is available.

In terms of the operation of a mobile robot, false negative recognition is not very serious. If a false negative occurs, then MARVEL will not be able to match the current data to a model in its database even though the correct model exists. A new room model will then be built from the current set of data and entered into the database. The robot will think that it is in a new room that it has never encountered before. This room model will be built up as recognition is performed against it, as will the other room models. A background process that tries to recognize model versus model will be running, however, to look for just this occurrence. If it finds two models that match. they will be combined and the world model compacted based on this match (see Section 10.4).

Although MARVEL is designed to avoid false positive recognition, it is always possible that this might occur. In this case, the mobile robot will think that it is in one particular room when it is actually in another. This will lead to inconsistencies in the connections between locations in the world model. For example, because of false positive location recognition, connection information in the world model could state that a particular doorway in Room A leads to both Rooms B and C. Because these inconsistencies are detectable, the possibility exists of correcting them. A local exploration of the area might yield enough structure of the inconsistent part of the world to resolve the problem. The robot could place a temporary beacon in one of the inconsistent rooms and then explore the confounded path with absolute confidence in recognizing the original room by the beacon. The most reliable strategy, however, would be for the robot to ask for help from its human owners to resolve the inconsistency.

One would expect more errors in recognition against the early models, which are supported by very few data sets, than in recognition against the later models that are based on many data sets. Testing confirmed that this is indeed the case. Figure 8-1 shows that the number of false negative recognition occurrences fell as the model used for recognition improved. (A matched model threshold of 50% and a matched data threshold of 80%, as described in Section 6.1, were used for the testing described in this below.) Examination of the 144 trials involving the initial models, as described in Section 8.1, shows that there were 33 trials with correct recognition possible and that only 9 of these resulted in a false negative result. This number dropped to 7 false

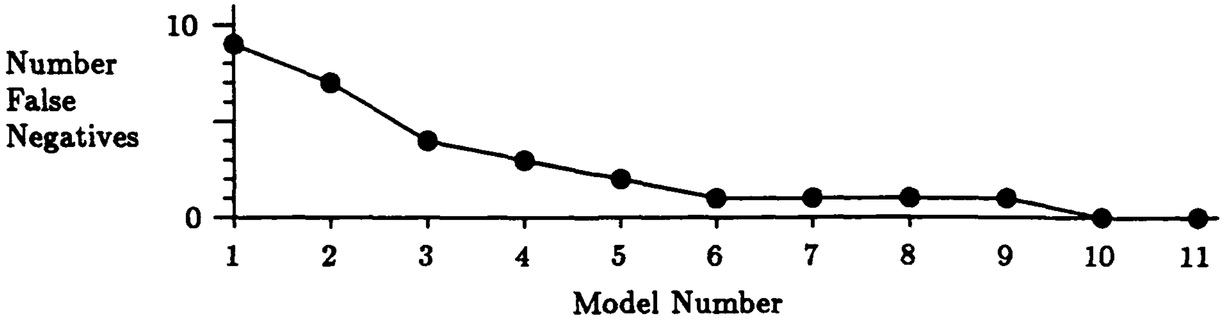


Figure 8-1: Graph of the number of false negative recognition occurrences versus the model used for recognition. Model 1 incorporates the information from one data set, Model 2 from two data sets, and so on.

negatives out of 30 correct possibilities for the 105 trials using the second models. This downward trend continued for the remaining trials with the evolving models. (The decreasing number of trials with correct recognition possible is due to the fact that data sets that were incorporated into a model were not checked against it in the recognition testing.)

The error *rate* can be examined by considering the percentage of trials with correct recognition possible that resulted in a false negative result. The percentages for the results above are shown in Figure 8-2. The error rate for the 9 false negatives out of 33 possible correct trials for the initial models was 27%. This rate dropped to 23% for the trials using the second models (7 out of 30), and continued dropping until reaching 10% for the fifth model trials. For succeeding models, the false negative rate stayed less than 10% except for the ninth model trials. The lower error rate for the fifth and subsequent models is to be expected when the specific parameters (rise time = 4, hysteresis = 4) of the model update algorithm used in these tests are considered. (See Chapter 7 for an explanation of the model update process.) The weights of the important model points increase steadily over the first several data sets, but the maximum model point weight cannot be achieved until the fourth model. In addition, no model points could be retracted until the fifth model. Thus, the fifth and subsequent models best reflect the rooms they represent.

The rise in the false negative rate toward the right side of the graph in Figure 8-2 is due to the smaller number of correct recognition trials possible. For example, a

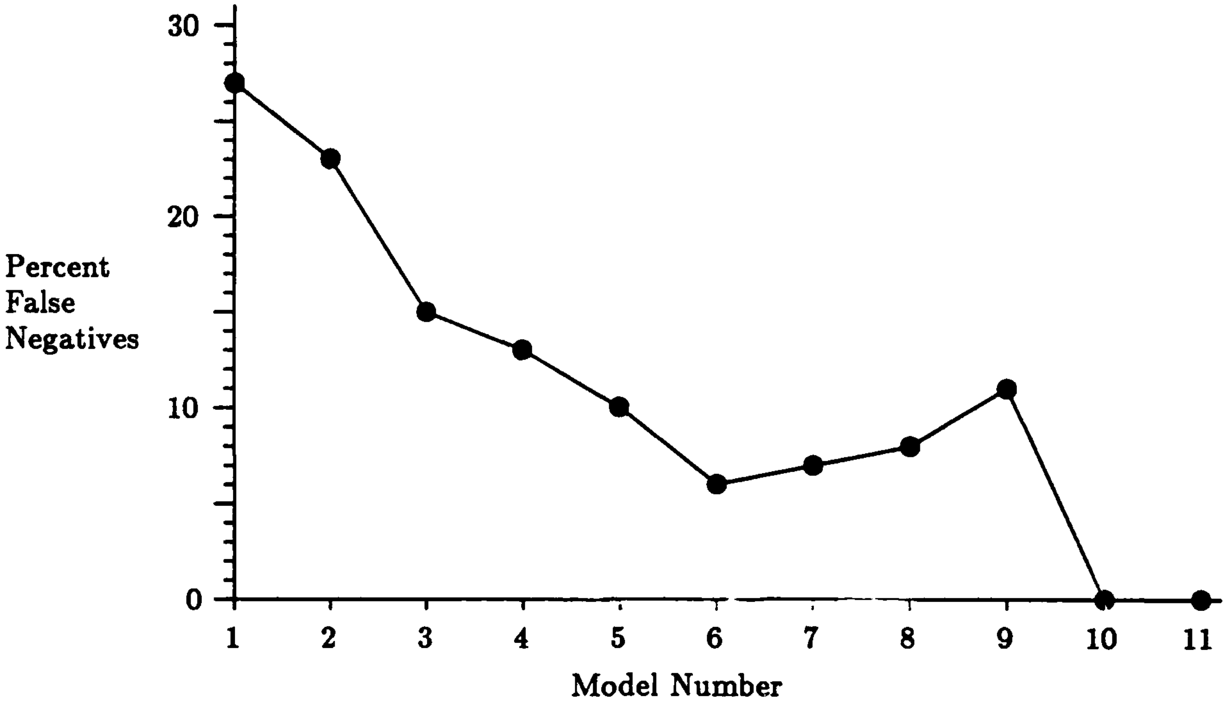


Figure 8-2: Graph of the percentage of false negativc recognition occurrences versus the model used for recognition, considering only trials with a correct model/data match. Model 1 incorporates the information from one data set, Model 2 from two data sets, and so on.

single false negative for the ninth model trials yielded a false negative rate of 11%. Examination of the recognition test results shows that the actual number of false negatives is low for when the built-up models are used (Figures 8-5 and 8-6): in the ninth model trials mentioned, the 11% false negative rate was due to only a single false negative recognition occurrence. The relatively large percentage is an artifact of the testing method. For the ninth models, only three data sets that have not been incorporated into the models remain for each room. Thus, the single false negative recognition occurrence out of nine possible correct recognition trials yielded an 11% error rate.

As the model and data thresholds are varied, the rate of false negative recognition also varies. The false negative rate increases as the model threshold is increased. This increase is expected, since the higher threshold makes it more difficult to obtain an acceptable match in the presence of noisy data using evolving models that do not exactly reflect the current state of the locations. Figure 8-3 shows this rate for three different model thresholds. With a model threshold below 37%, false positive recognition errors occurred. The false negative rate also increases as the data threshold is increased, which is expected for the same reasons as the increase related to the rise in the model threshold. This effect is shown in Figure 8-4 for three different data thresholds. Below a 69% data threshold, false positives occurred. The actual numbers of false negative recognition occurrences are shown in Figure 8-5 for the different model thresholds (corresponding to Figure 8-3), and in Figure 8-6 for the different data thresholds (corresponding to Figure 8-4).

If these recognition results are considered against *all* of the recognition trials that were performed, it is seen that the actual chance of false negative recognition is fairly small. Figure 8-7 shows this false negative rate. Based on these results, there is only a 2–3% chance that MARVEL will return a false positive recognition result when comparing a data set to the database of models.

For all of the trials reported in this section, no false positive recognition occurred. Thus, reasonable false negative rates can be achieved through a good choice of the recognition thresholds while still avoiding incorrect recognition of a data set with an

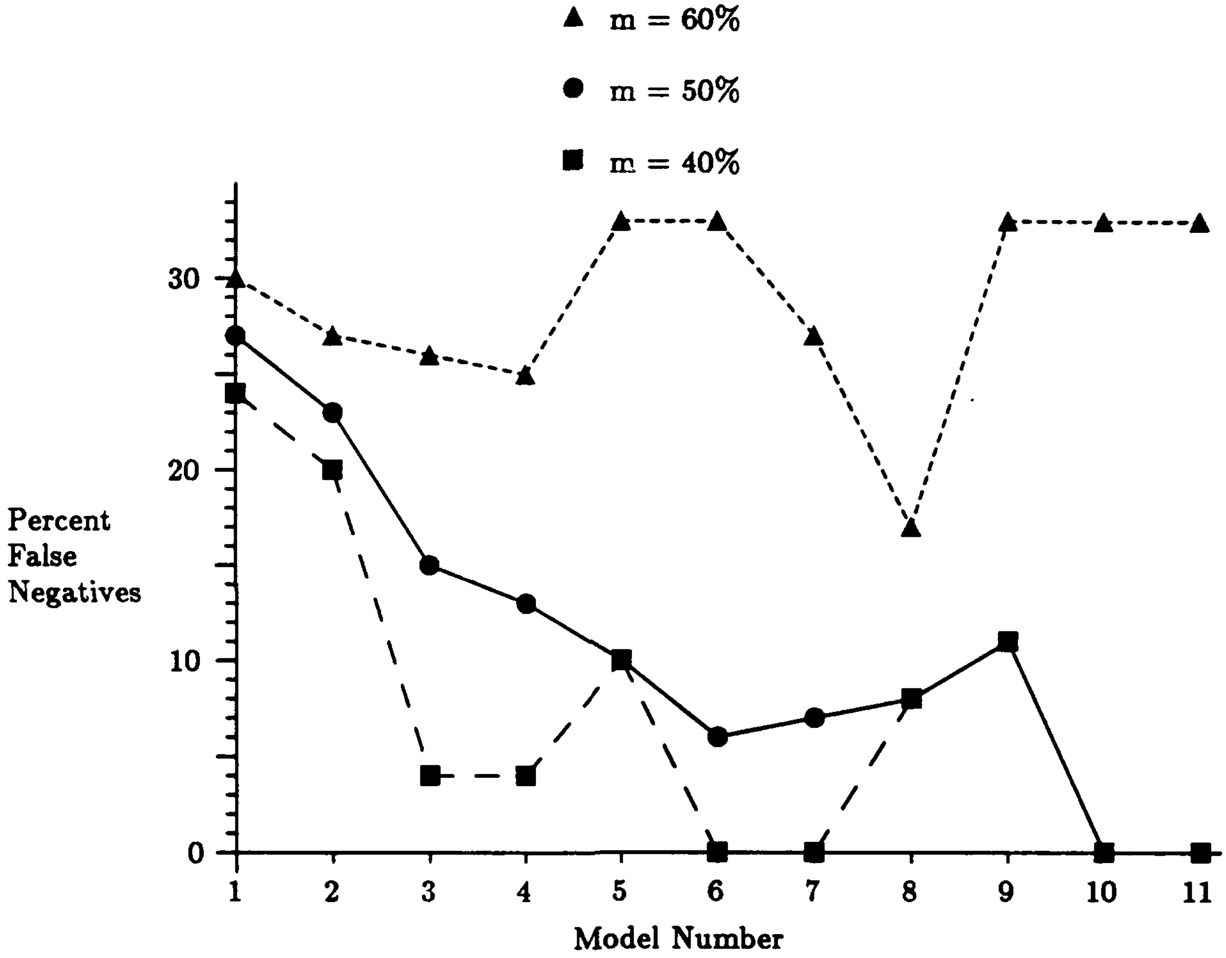


Figure 8-3: Graph showing the effect that varying the matched model threshold has on the percentage of false negative recognition occurrences, considering only trials with a correct model/data match. The data threshold is held constant at 80%. The square data points plot the 40% model threshold, circles 50%, and triangles 60%.

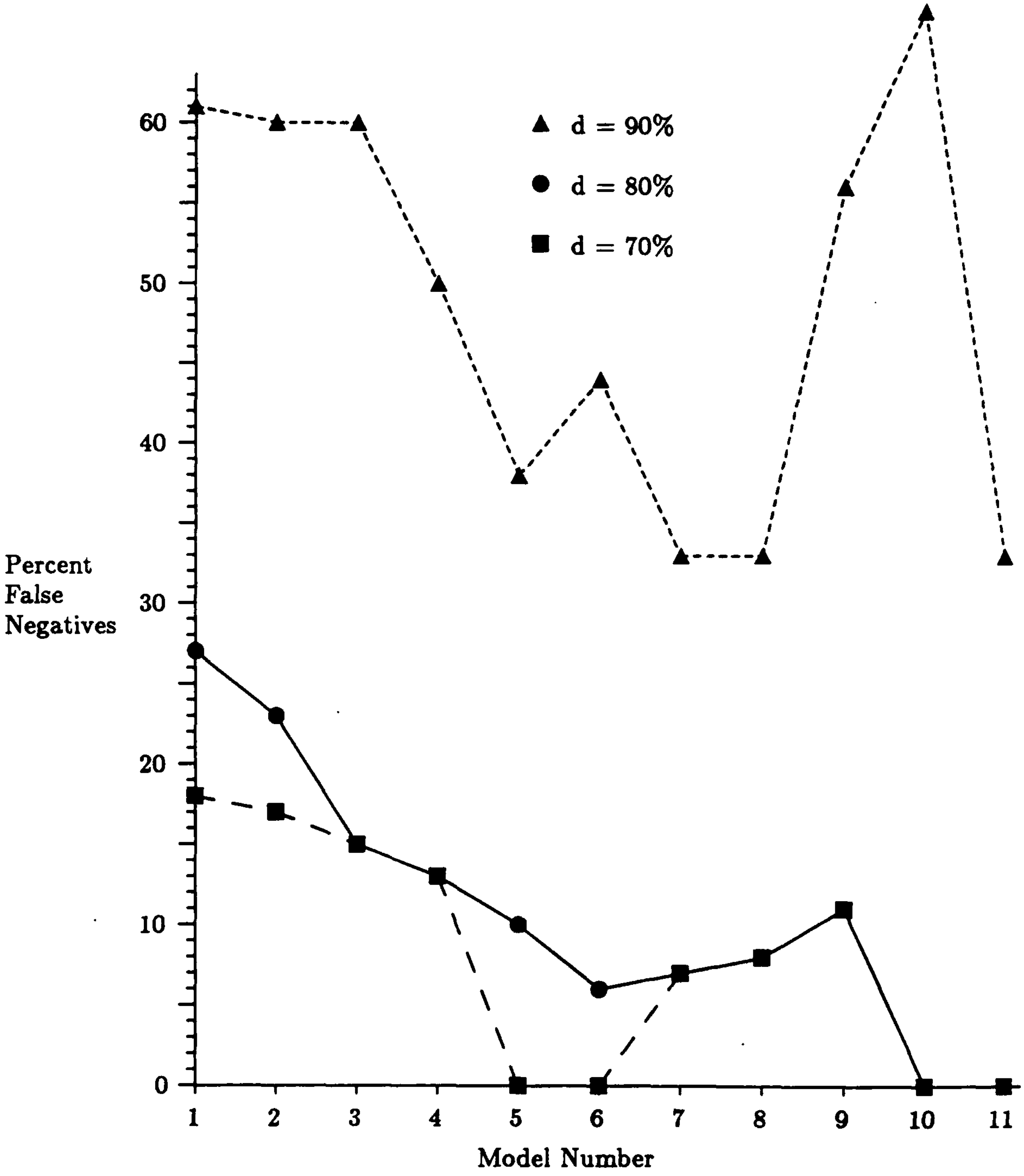


Figure 8-4: Graph showing the effect that varying the matched data threshold has on the percentage of false negative recognition occurrences, considering only trials with a correct model/data match. The model threshold is held constant at 50%. The square data points plot the 70% data threshold, circles 80%, and triangles 90%.

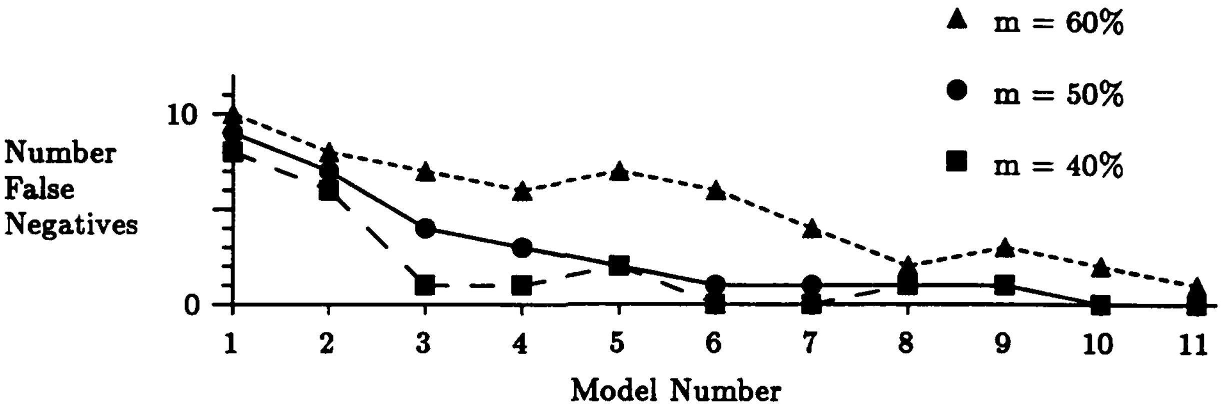


Figure 8-5: Graph showing the effect that varying the matched model threshold has on the number of false negative recognition occurrences, considering only trials with a correct model/data match. The data threshold is held constant at 80%. The square data points plot the 40% model threshold, circles 50%, and triangles 60%.

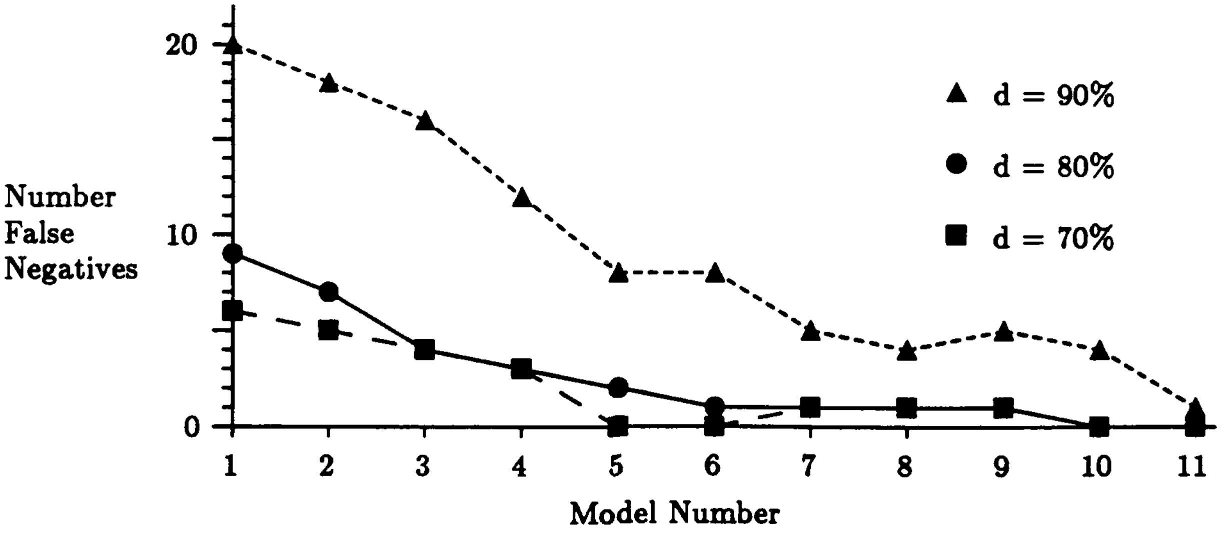


Figure 8-6: Graph showing the effect that varying the matched data threshold has on the number of false negative recognition occurrences, considering only trials with a correct model/data match. The model threshold is held constant at 50%. The square data points plot the 70% data threshold, circles 80%, and triangles 90%.

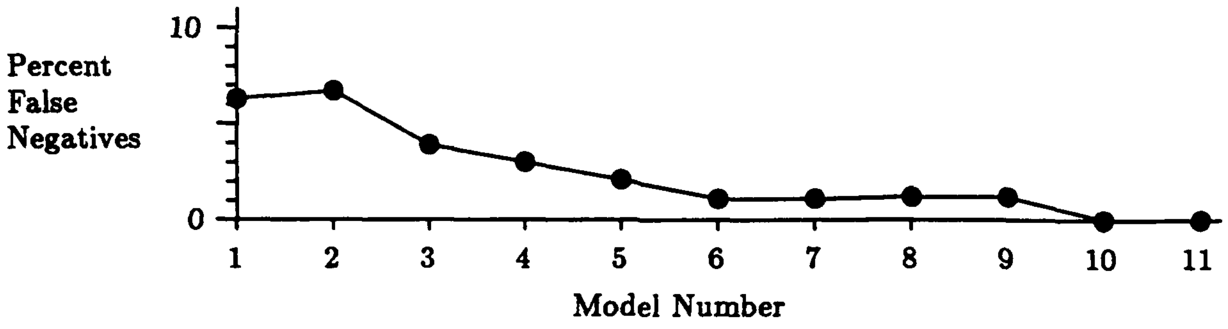


Figure 8-7: Graph of the percentage of false negative recognition occurrences versus the model used for recognition, considering all recognition trials. Model 1 incorporates the information from one data set, Model 2 from two data sets, and so on.

existing model.

8.3 Localization Performance

A secondary goal of MARVEL is to localize the position of the robot in the room that is recognized. By comparing the known positions of the robot during testing to the positions determined by the translations and rotations of the recognition transforms, MARVEL's localization performance can be judged. Localizations were achieved to within a distance of 1 foot and 10° of the true robot position for successful recognition trials. This performance is quite good when the grid size of the model/data representation and the inherent uncertainty of the stereo data are taken into account (see below). The localization accuracy is sufficient to bring other task modules (e.g., a door finder) to bear on an area of interest. Models incorporating four or more data sets yielded the best localization results. As was seen above in Section 8.2, these models produce fairly reliable recognition since the models have had a chance to build up the important recognition features. The heavier weights for these repeatably distinguishable features help in the localization performance.

The localization error can be attributed to a few different sources. First, the occupancy grid representation allows some variation in the transform that brings the model and data into correct correspondence. Because of the 1 foot grid width used for testing, the model and data points were only represented to within $\pm\frac{1}{2}$ foot of their observed locations. This finite resolution limits how precisely the points can be represented and thus how well they can be matched. The grid resolution, coupled with the 1 foot final search radius of the matching algorithm, implies that the input points that gave rise to a model point and a data point could be separated by up to 2 feet and still be matched. At a distance of 11 feet, that separation would allow a rotation of just over 11° between the model and the data. Second, there is some uncertainty associated with the stereo data itself (Section 2.3.3). At a distance of 15 feet, there is a maximum distance error of 2.5 feet in the determination of the true 3-D position of a stereo feature. Finally, due to the imperfect models and data, a complete match between the two cannot be attained. Counteracting these error sources is the effect of a large number of model and data points as well as the weights of the model points.

Because many features are used for recognition (and hence for determination of the position of the robot), the uncertainties in the positions of the individual features tend to average out. The effect of these uncertainties is also limited since data (model) features with large positional errors are typically not close enough to the corresponding model (data) features to allow a match and hence do not affect the recognition and localization results. In addition, the heavily weighted model points affect the transformation determination more than the lightly weighted ones. Since these points are known to be relatively reliable, their emphasis helps the performance of the recognition process. All of these effects counteract the localization uncertainty due to the match of any particular model and data point. The results show that using the aggregate of a set of simple features for recognition (instead of depending on a few complex features) yields good localization as well as recognition results.

8.4 Running Time

MARVEL was tested on a Symbolics 3600 computer. The only special hardware attached was a frame grabber with associated color display driver and a custom-built hardware convolver. The stereo algorithm took 1 minute with the hardware convolver and 2 minutes, 28 seconds without it to process a pair of 288 × 227 pixel stereo images. Most of the running time of the stereo algorithm is devoted to finding the edges in the images—matching the edges took less than 12 seconds. A Canny edge detector [Canny 1986] was used with the hardware convolver. Without this special-purpose hardware, the edge detector of [Deriche 1990], a recursive implementation of the Canny algorithm, was used. As described in Section 2.3.5, by using an accurate stereo camera turntable only eight stereo pairs are needed for full 360° coverage around the robot. Thus, 8 minutes are required to obtain the input data using the hardware convolver, while 19 minutes, 44 seconds are needed using only the serial machine. (The cameras can be turned while the stereo algorithm runs, so no time is lost there.) A stereo algorithm implemented at the MIT Artificial Intelligence Laboratory on the Connection Machine, a SIMD parallel computer, takes approximately 5 seconds to process a stereo pair [Drumheller and Poggio 1986] [O'Donnell 1990], thus requiring only 40 seconds to obtain the complete coverage of stereo input. A different possibility for a speedup of the stereo algorithm is presented in Chapter 12, based on the fact that only 2-D stereo data is used to build the representations.

The recognition process takes 84 seconds to check a data set against a model. Most of the processing time was spent performing the array operations of the least-squares minimization process. Instead of writing optimized code, the Symbolics array utilities were used for the least-squares minimization. A hardware speedup for this minimization is discussed in Chapter 12.

8.5 Effect of Grid Size

Chapter 4 describes the occupancy grid representation used by MARVEL for both models and data. For the testing that produced the results described above, grid squares 1 foot on each side were used. Other testing was performed with 1/2 foot, 2 foot, and 3 foot grid squares to investigate the effect of grid size on the recognition algorithm.

As seen above, the 1 foot grid size yields an acceptable false negative rate while eliminating false positives. Recognition with the 1/2 foot grid size had roughly the same false negative rate. There was less abstraction of the stereo data with this smaller grid size, however, and the running time rose to 108 seconds due to the larger number of model and data points. The recognition time fell to 49 seconds with a larger grid size of 2 feet. At this size, the false negative rate rose to approximately 30% and the localization performance deteriorated to roughly 2 feet and 20°. Recognition was even faster using the 3 foot grid, taking only 30 seconds. This grid size was unusable, however, due to poor recognition and localization performance.

8.6 Response to Changing Location Features

As mentioned in Chapter 1, MARVEL is intended for use in a world that experiences change. Its response to change was explored in two ways. First, the data sets for the various rooms were collected over the course of six months, so that changes that naturally occurred over this time period were incorporated into the data and models. These changes can be seen in the differences from model to model in the figures in Appendices C, D, and E. Second, a specific room feature was added and later moved to a new position so that the action of the model update algorithm could be observed. This test is described next.

A large crate was placed in one corner of Room 914 after a few data sets were taken. Later during the testing, this crate was moved to a different position. The model points that correspond to the crate are shown circled in Figure 8-8. The evolution of the representation of the crate can be seen in Models 4–8 (Appendix D, Figures D-4 and D-5). The box was then shifted a few feet back into the corner of the room. The corresponding new model features are shown circled in Figure 8-9. The build-up of this new position and the fade-out of the old one can be seen in Models 9–12 (Appendix D, Figure D-6). This example demonstrates that the models successfully respond to changing room features. New model points are added and their weights increased for new location features, while model points that no longer correspond to location features are eventually retracted from the models.

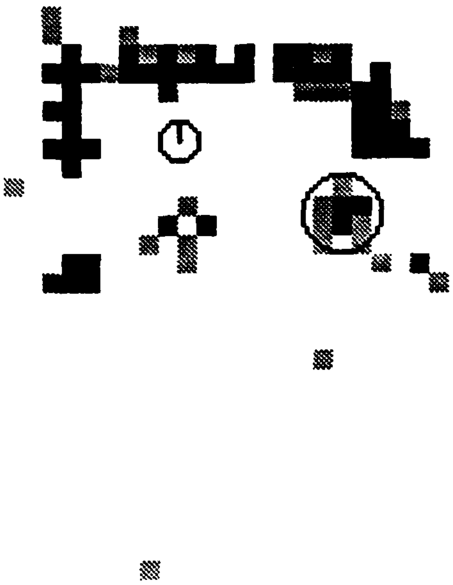


Figure 8-8: Room 914 Model 8 showing the model points corresponding to the original position of the crate that was placed in one corner of the room (circled).

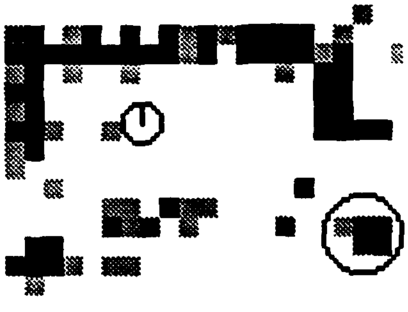


Figure 8-9: Room 914 Model 12 showing the model points corresponding to the new position of the crate that was placed in one corner of the room (circled). Also note that the model points corresponding to the old position have mostly faded out.

The gradual build-up and fade-out of model features is controlled by the rise time t_r and hysteresis h parameters of the model update algorithm. The rise time controls how quickly a new feature is accorded importance in the recognition process and how quickly the importance of an old feature is decreased. The hysteresis controls how sensitive the update is to the disappearance of model features. These two parameters combine so that a fully-weighted model point will not be completely retracted until it has not appeared in $t_r + h$ data sets. On the other hand, if some movable feature in a room regularly appears in one of two places, this retraction delay will tend to keep both feature positions in the room model. The effect is much like the persistence of perception that the human eye experiences: features that go rapidly into and out of perception will persist in the models. Thus, the acceptable rate of change of room features is proportional to $\frac{1}{t_r+h}$ for the full incorporation or removal of points from the models. Partial effects are seen over correspondingly short time periods.

MARVEL responds to a slowly changing world by altering its location models. Relatively stable location features become heavily weighted in the models, as can be seen by the heavily weighted model points in the figures in Appendices C, D, and E. Model features are retracted if their corresponding location features disappear (e.g., the crate no longer in its original position). Finally, as new location features appear, they are gradually added to the models (e.g., new crate position).

8.7 Response to Variations of the Model Update Algorithm

The model update algorithm described in Chapter 7 contains two free parameters: the rise time to full weight and the hysteresis (delay) for reducing the weights of the model points. The rise time affects the influence of new versus old model points, with larger rise times favoring the older model points. The hysteresis determines how old the information that last contributed to a model point can be before that model point's importance is lessened. For the testing results described in this chapter, the linear model point weight update was used, with a rise time of 4 and a hysteresis also of 4. In addition, the model update algorithm was tested with both the linear and exponential weight update functions for rise time = 1–8 and hysteresis = 1–5. Appendix G shows the final model for Room 914 (Model 12) for each of these possibilities using the linear function, while Appendix H repeats these models using the exponential function.

Comparing the models of the different update functions, it is apparent that the newer points are more lightly weighted for the exponential update than for the linear update. Given that difference, the following comparisons apply to either method. In the models with low hysteresis, some room features are not well-defined. On the other hand, a high hysteresis leads to persistence of features that no longer exist in the room. It is also apparent from these examples that longer rise times lead to build-ups and fade-outs of changing model features. Through all the update possibilities, however, the overall form of the models remains the same even though individual details differ. The choice of a rise time and hysteresis in the range 3–5 yielded reasonable results for this room. In general, the choice of these parameters should be based on assumptions about the world and the input process (Section 7.2).

8.8 Free Parameters

MARVEL uses several free parameters in its various algorithms. Some of these parameters are used to make the algorithms more efficient, e.g., search cutoffs, while others affect the final results of the algorithms by changing the representations used. A complete listing of the free parameters is given here. This listing should be useful for researchers who wish to build MARVEL for their own applications.

The grid-square representation involves the choice of the size of a grid square, as mentioned in Section 4.3. Results of testing MARVEL with various grid sizes are given above in Section 8.5. The choice of a grid size for the representation involves a tradeoff between faster processing for fine grids and lower sensitivity to the uncertain positions of data features for coarse grids.

The determination of the initial alignments in Section 5.1 involved finding linear groupings of model and data points, taking pairs of these groupings, and matching model pairs with data pairs. This matching could be performed for all possible groupings and all possible pairs. To speed up the processing, however, consideration is limited to the groupings and pairs that are most likely to yield correct alignments. The lines are ranked by the number of points they contain and the ten lines with the most support are chosen. Pairs are formed from these lines, ranked according to the angles they form, and the ten most perpendicular chosen. This selection of the top ten in each case was based on the observation that the best choices typically occurred in the top five ranked lines and line pairs.

The matching of line pairs also involved a determination of how well and how reliably two pairs matched. The matches are investigated in a ranked order, so the cutoff parameters for this determination are ancillary to the main algorithm. Since these cutoffs simply help limit processing to those line pairs that are most likely to yield good alignments, they can be eliminated entirely without affecting the results that MARVEL produces. The 20° minimum angle for a pair of lines was chosen arbitrarily to restrict the lines paired to those that have stable intersection points. The pairs that resulted in correct alignments typically had angles of greater than 45°. Due to noise, a model line pair and a data line pair will rarely have the same angle. The requirement that the pairs agree within 10° simply puts a bound on how closely they must agree. Testing showed that the angles of correctly matched pairs were always within this limit. The feasible match check was based on model and data point sets having eccentricities greater than .7 and major axis alignments better than 45°. Again, this is a quick test to eliminate obviously bad matches and can be left out entirely.

The lines that form the pairs used in the initial alignments were found through a Hough bucketing scheme (Section 5.1). As with any such bucketing, the size of the buckets must be chosen. The bucket size determines how close to colinear the two-point lines must be to contribute to the same Hough bucket (and thus to the line that the bucket defines). In addition, buckets that are too small will contain few, if any, lines, while buckets that are too large will cluster lines that are not closely related. The choice of a 1.5 foot $\times$ 5° bucket size was determined heuristically to balance these concerns. The grid nature of the data can lead to a preference for lines along the grid rows and columns, so a minimum distance of 1.1 foot between data points that form a line is enforced to avoid this. (The 1.1 foot minimum distance is 10% greater than the 1 foot spacing between grid squares.) The 10 foot maximum distance between data points that form a line gives a rough locality measure so that accidental alignments of distant points are largely ignored. The line-defining parameters can be relaxed at the expense of the generation of more lines to consider.

The least-squares transform refinement of Section 5.2 uses a set of search radii to find matching model and data points. The smallest radius of 1 foot was chosen to match the spacing between the grid square centers. The larger radii of 2 and 3 feet were chosen at multiples of this grid spacing. Improvement of the transform is judged by considering the transform variance, S^2. In general, this variance cannot be reduced to zero, so an end condition on the iterative improvement is required. If the variance increases at any step of the improvement iteration, the process should obviously be stopped. Otherwise, limits were chosen so that the attainment of a small enough value of the variance or a small enough change in the variance will stop the transform improvement cycle. The limits of $S^2 < 1$ and $\Delta S^2 < .1$ were chosen by observing the change in the model/data alignment and selecting values that yielded no visible alignment improvement.

The percentages of model and data points matched are used as recognition thresholds, as described in Section 6.1. These thresholds were selected to yield no false positive recognition occurrences while keeping the false negative rate acceptably low. Section 8.2 discussed the sensitivity of the recognition results to these thresholds.

Finally, the choice of a rise time t_r and hysteresis h affect the results of the model update algorithm (Section 7.2). The choices of these parameters influence the responsiveness of the models to new data as opposed to existing data. The results of testing the model update algorithm with various values of these parameters, as well as with two different weight update functions, was reported in Section 8.7.

Within reasonable ranges, the choices of these free parameters in MARVEL do not affect the correctness of the recognition results. Instead, they affect performance tradeoffs in the system. Some of the thresholds balance the level of abstraction against the accuracy of representation in the model and data. Others are search cutoffs, which gain search speed at the expense of the completeness of the search. In general, the parameter tuning is used to obtain optimal performance of the system. Reasonable parameter values based on knowledge of the algorithms and problem domain, however, will suffice to yield good results.

8.9 Summary

This chapter described the testing used to demonstrate that MARVEL successfully supports mobile robot navigation through recognition of world locations. Specific testing data was presented to demonstrate that MARVEL effectively recognizes locations based on the models that it has built and determines its position in those locations based on the recognition results. The false negative recognition rate is kept acceptably low while still eliminating false positives. The localization is sufficiently accurate that MARVEL can be used to drive the robot close to an area of interest so that other recognition and task modules can be used to position the robot exactly.

Running times were presented for the stereo and recognition algorithms. These times are reasonable for the proof-of-concept work that this thesis describes. Possibilities for speeding up the implementation are discussed in Chapter 12. The sensitivity of MARVEL to variations in the recognition and model update parameters was also considered. Although a range of choices for these parameters provides reasonable results, the testing shows that the system performance can be tuned through the parameter value selections. The evolving model for Room 914 was also examined and found to have a good response to changes in the room features.

To this point, the thesis has discussed MARVEL and its focus on recognizing world locations. The next chapter describes a world modeling scheme with which MARVEL is well-suited to work.

Chapter 9

Location Recognition and the World Model

The location (room) models described above are part of a larger world modeling system under development. The world model represents the known world of the robot and contains information that allows the robot to navigate between different locations. This chapter describes the world model and how it uses the location recognition ability of MARVEL.

The mobile robot's world will be represented by a topological map similar to that described by [Kuipers 1977]. The world will be represented by locations and information about how they are connected. Rather than being supplied *a priori*, this world model will be built by the robot as it explores its environment. Because the robot has long-term autonomy, existing for days, weeks, even months at a time in its environment, the amount of time incurred in building the initial world model is amortized over the lifetime of the robot. Because the robot builds and maintains its own world model, it can react to a (gradually) changing world in a way that a robot depending on a precomputed map cannot. Thus, the world model, coupled with the location recognition system, will allow a mobile robot to go from its current location to a desired new location in the much same way that people use maps and books (e.g., [Baxandall 1983]) to plan their daily commutes or vacation excursions.

9.1 Introduction

Much of the current research in navigation planning and world modeling for mobile robots concerns *mission-level autonomy*, i.e., autonomous operation over a limited time frame for the accomplishment of a specific goal. This chapter is concerned with a world model to support *long-term autonomy*, i.e., autonomous operation of a mobile robot over a period of days, weeks, or months for the continued achievement of a set of goals. For example, a mobile robot was designed to "live" in the MIT Artificial Intelligence Laboratory, continually collecting empty soda cans and returning them to a central repository [Connell 1989].

In mobile robots with mission-level autonomy, the world model acquired while on the mission is typically limited. The world model need only be extensive and detailed enough to allow the accomplishment of the mission's goal. Any further effort put into building a more extensive world model than is needed to achieve the goal would be wasted. Obviously, a robot autonomous at the level of a single mission cannot take advantage of what it has learned about the environment to achieve further goals since there are none. Thus, common sense dictates that only the minimum amount of effort necessary for successful completion of the mission can be devoted to representing the environment. On the other hand, a mobile robot that must exist in its environment for long periods of time to accomplish a number of goals can afford to devote more of its efforts to modeling the world. Although acquiring the world knowledge may involve a large effort initially, the resulting store of knowledge can be used throughout the lifetime of the robot, enabling it to achieve its goals more efficiently overall than if it had not built an extensive world model.

Once the world model has been constructed, only a relatively small amount of effort is required to update this model as the robot goes about its business. The largest part of the world modeling task involves building the structure of the model and filling in the descriptive information. Updates to the model entail only local modifications to the structure and descriptions, which are much less costly than the initial construction of the model. When the effort to build the model and update it over time is amortized over the life of a long-term autonomous mobile robot, the average amount of effort during the robot's life that is devoted to the world model is fairly small.

An autonomous mobile robot must be able to maintain the world model, however, in the face of sensing error, its own positional uncertainty, and limited resolution of both the sensors and the representation. Other researchers have addressed the problem of maintaining the consistency of a world model as a robot moves through its environment [Brooks 1985] [Chatila and Laumond 1985] [Faugeras *et al.* 1986] [Moravec and Elfes 1985]. Due to the errors ar' uncertainties mentioned above, the difficulties encountered are usually a direct result of an attempt to preserve exact metrical knowledge about the world. This chapter describes how qualitative world knowledge (instead of exact metrical information) can be used to build a consistent world model that supports navigation planning.

One of the most interesting problems involved in the creation and use of such a qualitative world model (and indeed in the use of any world model) for navigation is the need to recognize previously-encountered world locations. Practical considerations such as finite world model resolution and errors inherent in any sensing of the world lead to the conclusion that no world model can be followed indefinitely without eventually recalibrating, i.e., comparing the robot's current position to the model and either verifying the position or, failing in the verification, finding the current world location in the world model. Thus, a world-location recognizing system such as MARVEL is essential to any navigation scheme that uses a world model.

9.2 Navigation Planning

Mobile robot navigation is the locomotion of a robot through an environment without collision in order to achieve some goal. Navigation planning, then, consists of two distinct facets: planning routes through the world to achieve a goal and planning paths through the environment immediately around the robot to avoid obstacles.

Navigation planning on a global scale is route planning. Routes are planned between specific locations in the world and are described in terms of the locations that they encounter and the pathways that they traverse. For example, the route from the bedroom to the study might be described as follows: Go out the bedroom door and turn right. Proceed down the hallway and into the living room. Go through the door at the far end of the living room to enter the study. This route planning can be performed offline before the robot begins a trip.

When traversing a route such as described above, a robot must avoid obstacles, e.g., chairs, tables, people, etc. The exact path around these obstacles cannot be planned ahead of time since the locations of these objects can change. Thus, the planning of paths around specific obstacles is left until the obstacles are actually visible (to the sensors). This planning "of the moment" is also used to refine the planned route by taking into account the robot's current location and the world that it perceives. For example, the robot may know where to expect a doorway, but to pass through it the robot adjusts its heading according to where it actually perceives the doorway.

9.3 Requirements on a World Model

From the discussion above, it is evident that global features of the world must be represented to support route planning as well as local features to support path planning. Since the types of planning to be supported and the world knowledge they require are of different scale, a two-level representation is appropriate.

For global route planning, a robot depends on a stored model of the world. Therefore, the global world model should represent information that is *invariant* under typical world perturbations. Also, for robustness the information represented should be invariant with respect to sensing modality and sensor precision. Global route planning depends heavily on the invariance of *connectivity* of locations in the world. Connectivity information is typically acquired experientially. For example, a path would be known to exist between A and B because the robot had previously traveled between these two locations. Route planning also requires information about *order-*

ing, e.g., that location B lies between locations A and C along a particular path. *Directional information* is also needed, along the lines of "When at location A and with a particular orientation, location B lies to the left."

Exact metrical information is neither required nor even desirable. Since all distance measurements are subject to error and are bounded by the limitations of sensor accuracy, distance measurements are best expressed in qualitative terms such as "medium far," "short," etc., where terms such as these represent overlapping ranges that can confidently be assigned to measurements even in the presence of uncertainty. The same sort of arguments apply to exact direction headings. Realistically, no matter how confidently a robot expects that a corridor lies 90° to the left, it should go down the corridor lying closest to that direction rather than run into a wall.

To help identify locations and determine the robot's position with respect to the locations, a recognition model is associated with each location in the world model. These recognition models allow the robot to check its progress along its planned route and to orient itself near known locations. The model building and recognition capabilities provided by MARVEL serve just this purpose.

For local path planning, a stored memory of the world should not be depended upon since the exact locations of obstacles can change. Instead, paths are planned around obstacles in the immediate area of the robot based on its perception of this area. Thus, only a limited region around the robot need be represented accurately, noting the locations of the robot and nearby obstacles. In this way, the problem of maintaining the consistency of this accurate positional information with respect to other regions of the world model is explicitly avoided. A possible approach to local path planning is outlined in Section 9.4.2.

9.4 The World Model

Locations, paths, features, etc., have been mentioned with respect to the world map. How should these or other pieces of information [Lynch 1960] [Kuipers 1977] be represented? Local, viewer-centered information has also been mentioned—what of its representation? A two-level world model is described below that incorporates this large-scale and small-scale information and that supports both the global route planning and local path planning facets of navigation planning.

9.4.1 The Topological Map

The part of the world model that represents large-scale space is the *topological map*. The topological map contains information about how the different pa:ts of space are related and what they contain. This information allows the robot to plan routes through the world and verify its position as it travels along them. The topological map is built by the robot as it explores the world and is based on information that the robot has gathered about the world.

The topological map is a simple graph structure consisting of *locations* and *connections* (Figure 9-1) [Kuipers 1977]. A location is a distinguished place in the world and has finite spatial extent. The position of a location is determined by its connections to other locations. A connection is simply a navigable transition between two locations. Locations have labels, recognition models (from MARVEL), and other information associated with them.

Locations can be determined as salient or important locations in the environment in a number of ways. Locations wnere a change in direction of travel is or can be made are obvious candidates for locations, e.g., corners and intersections. Choice points such as intersections are obviously important for navigation since they mark a transition out of a homogeneous region. Intersections are also important since separately known paths are related via their common intersection. Transitions are features in the world that demarcate locations; for example, a doorway or a transition from a narrow corridor to a large open area separates two different locations. Locations can also be identified by their importance to a given task.

Different locations can be distinguished by their different characterizing features. Crossing a typical transition, e.g., passing through a doorway or rounding a corner, causes a sudden change in the perceivable features around the robot. Thus, the robot can determine that it has entered a new location because it will be presented with a

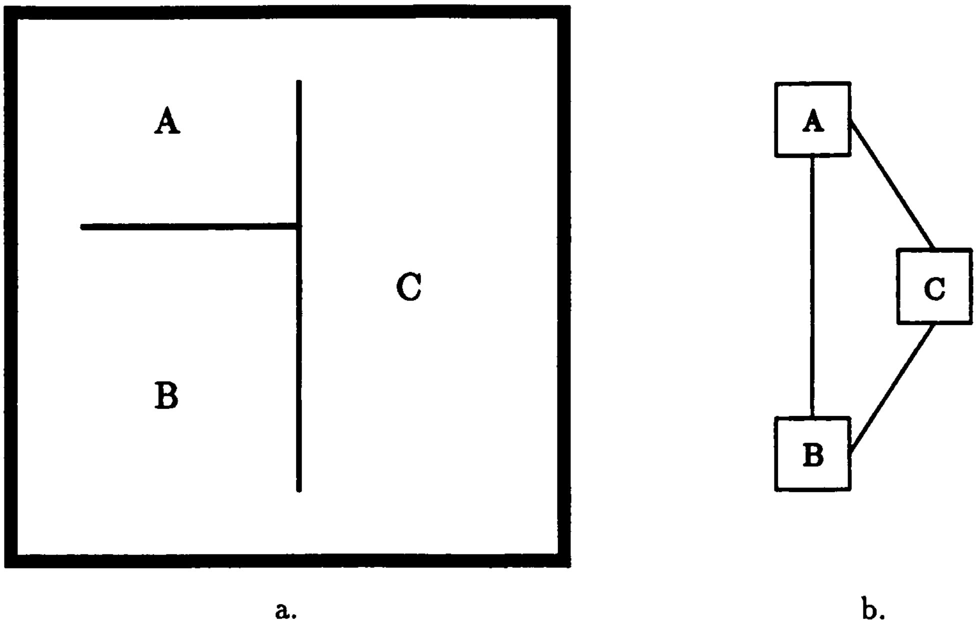


Figure 9-1: The topological map. a. A sample world. b. The topological map for the sample world. Squares denote locations and lines denote connections.

new set of features. Other sensor systems could also be used to determine if the robot has crossed such a transition. For example, sonars pointing out from the sides of the robot could report if it has passed through a narrow doorway (into another room).

Connections between locations are known because they are *experienced.* That is, a mobile robot will know that a connection exists between two locations by virtue of the fact that it has traveled between those locations. Information about a connection, such as the locations it links, its position in each of those locations, and the direction of travel through the connection with respect to a linked location, is acquired as the robot traverses the connection and is stored in the representations of the locations linked by the connection.

The topological map allows the problems of accumulated error in a world model to be avoided. This is accomplished by storing qualitative, rather than quantitative, information in the topological map. Connectivity relations between locations in the world are explicitly represented, but distance and direction data are stored as ranges, reflecting the uncertainty inherent in these quantities. The uncertainties about world locations that result from cumulative error are resolved by using MARVEL's recognition system to verify the robot's location with respect to the topological map.

Finally, the topological map allows the robot to overcome the *loop* problem: How can a robot travel away from a starting point so that it is no longer visible, then return to that point via a different route, while building a world map that has the starting point in correspondence with the final point? Cumulative error makes this problem extremely difficult if the world representation being built is an exact model of the world. Even if a recognition system is used to supplement an exact model, there is still a problem with exact models. Suppose the robot attempts to travel a loop and eventually arrives at a point that it recognizes as the world location from which it started. If, due to cumulative error, the model location for this point does not correspond to the starting point, the exact metrical information of the world model must be changed to make the two points coincide. There is then the decision of which stored measurements to alter. The "ripple effect" must also be dealt with as these alterations affect the metrical relationships between locations that lie along the loop path and other world locations. In using the topological map, however, the robot explicitly depends on MARVEL's recognition system to aid in determining its location. The connectivity information in this world model is not affected by cumulative error since this sort of error is of a metrical, not topological, nature. Working world modeling systems reported in the literature have all operated in essentially "open fields," that is, areas where all parts of the world are visible from any part of it (e.g., [Moravec and Elfes 1985]). Systems have been described that can navigate and model a loop in the world [Chatila and Laumond 1985], but it is not clear that the implemented systems have ever successfully done so.

Location Recognition

The robot needs to be able to identify known locations as it travels through the world. MARVEL supplies this capability through building, maintaining, and using world location models for recognition. After the topological map has been built, the problem of recognizing a location is simplified through the use of a very important constraint—the robot has an estimate of its location. Thus, the location model features and the robot's position with respect to the paths just traversed serve to identify a location. The model lets the robot determine when it has reached the location to which it is traveling; they are rarely needed for independent location identification.

Location models can, however, be used by the robot to locate itself if for some reason it becomes lost or confused. If the current location data does not allow a unique match with one of the known models, the topological map itself can be used to determine the current location: by moving from its current location and finding other locations and their features, the robot can build up a small map of the local area. This local map can be matched against the world map to determine the robot's location, with additional information added to the local map through exploration until a unique match is determined.

Since the topological map will be used to aid in location recognition, it must be possible to associate recognition models with each location in the map. The models will be added to the map as the map is built, used to identify locations as the robot moves through previously-mapped parts of the world, and updated as the world changes or additional information about various locations is acquired. The topological map, then, will contain information about individual world locations as well as the relationships between locations.

Since stereo features describe the existence of and relationships between salient parts of the visible environment, they also characterize world locations. Thus, stereo data is used by MARVEL to recognize locations. This data can also provide sufficiently detailed information to accomplish local path planning.

9.4.2 The Viewmap

The most detailed part of the world model is the *viewmap*, which describes as accurately as possible the environment immediately around the robot as perceived by the sensors. The viewmap is used to plan a path for the next motion of the robot. Because this motion may involve some backtracking, part of the area that the robot has just traveled through, but that is no longer visible, is kept in the representation. No attempt is made to store the whole world in this detailed representation, however:

only the world immediately surrounding the robot is considered. Thus, the problems of cumulative error are kept to a minimum. Because the viewmap is not stored the next time the robot comes to the same area, obstacles can change position between visits to an area without affecting the integrity of the representation. The design of this representation reflects the fact that although the robot can plan a general route based on its recollection of the world, the planning of a specific trajectory must depend on the current state of the world immediately around the robot [Braunegg 1989b].

The viewmap is constructed from 3-D information received from the stereo vision system (Chapter 3). The following discussion assumes that three-dimensional information of the area is required for performing some of the robot's non-navigational tasks. If this is not the case, the projection of the three-dimensional information onto a two-dimensional ground plane will suffice. First, the matched stereo segments are triangulated in the image plane without regard to their depth, ensuring that each stereo segment appears in the triangulation. Next, the scene is interpolated in depth based on this triangulation. Finally, following the work of [Moravec and Elfes 1985] in two dimensions and [Stewart 1988] in three dimensions, space is segmented into cubes by imposing a three-dimensional grid on the area to be represented. All cubes lying between the three-dimensional triangular patches and the viewer are declared to be freespace and the cubes occupied by the patches are declared to be non-freespace. Unmarked cubes represent unknown regions. (See Figure 9-2.)

After a movement of the robot, another view of the world is obtained and positioned in correspondence with the previous one (via the stereo features) to determine the translation and rotation of the robot. The freespace and non-freespace found from this additional view are then combined with the information obtained from the previous views to produce a more complete map of the area around the robot. Accumulated error due to the inexact determination of the movement of the robot is minimized by limiting the viewmap's domain to the area immediately around the robot.

The information obtained from stereo is subject to missing and extraneous data.

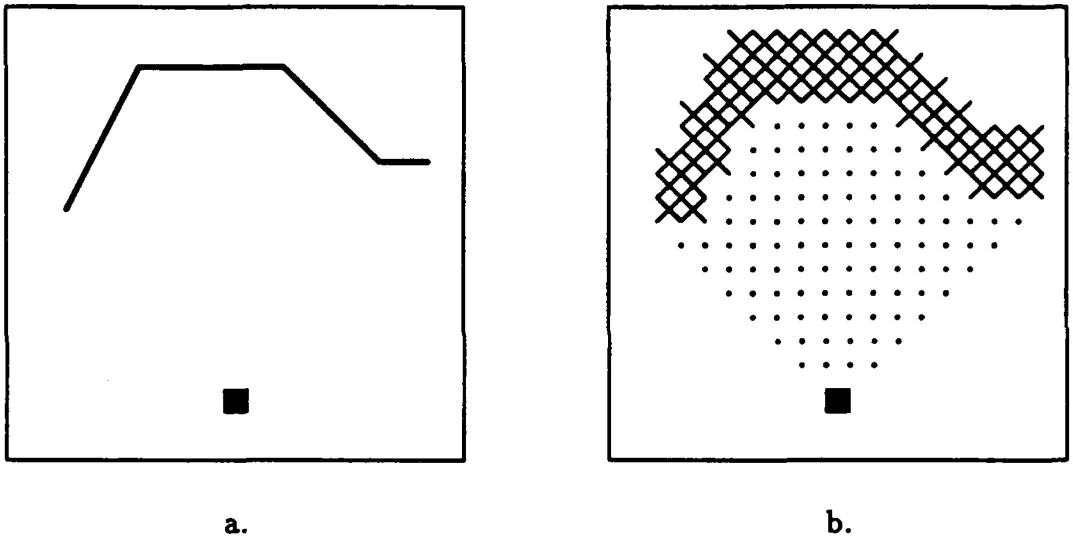


Figure 9-2: The viewmap. a. Surfaces (denoted by lines) perceived by the robot (square). b. The viewmap representing freespace (dots), non-freespace (X's), and the robot (square).

These errors are handled by marking freespace and non-freespace cubes with *confidence* levels. When updating the viewmap, overlapping freespace or non-freespace regions have their confidences reinforced (increased), while information from the current view that contradicts the (non-)freespace labeling stored in the map reduces the confidences in the labels. Stereo data also has an inherent positional uncertainty that can be represented by labeling non-freespace cubes in a region around the stereo data according to the error bounds obtained from the stereo geometry.

Once constructed, the viewmap contains sufficient information to plan paths in the visible area near the robot. Obstacles are identified by marked cubes while freespace is indicated by empty ones. The occupied status of the cubes can be updated as the robot moves along its path and the viewmap need only be maintained for a relatively small area around the robot. (As stated earlier, cumulative error precludes maintaining this sort of metrical map over large regions of the world.)

9.5 Summary

This chapter described a world modeling scheme that is supported by the location recognition ability of MARVEL. The two different aspects of navigation planning were discussed: global route planning and local path planning. These two types of planning have different requirements on the representations they use. The requirements on global route planning led to the choice of a topological map to represent the large-scale world. This map consists of world locations and the connections linking them. Exact metrical information is avoided in this representation to obviate the cumulative error problem. The topological map can be followed by the robot as it travels through the world, using MARVEL to verify its location with respect to the map. For local path planning, a viewmap is used. The viewmap represents detailed metrical information, but only in a limited area surrounding the robot. The topological map and the viewmap, coupled with MARVEL's ability to model and recognize locations, form a complete world modeling system that supports autonomous navigation and exploration of a mobile robot through its world. The next chapter describes how the mobility that a mobile robot provides can be used to advantage in MARVEL's model building, maintenance, and location recognition.

Chapter 10

Mobile Robot-Based Strategies

MARVEL is designed to support the navigation of a mobile robot through recognizing world locations. One can take advantage of the fact that the sensing platform is *mobile* to achieve recognition as well as for building and maintaining the location models. Several strategies for accomplishing these tasks using the mobile robot are discussed below.

10.1 Several Data Sets to Build Initial Models

As was seen in Chapter 8, the recognition performance of MARVEL increases as the models are built up over time. The greatest improvement in performance occurs over the first several model updates. Rather than letting these updates occur over time, the robot can build better initial models by taking several data sets the first time it encounters a room.

If several data sets will be obtained to build an initial room model, the robot should be positioned near the center of the room to acquire the first data set. (The room center can be estimated by considering the stereo data or input from other onboard sensors, such as a ring of sonar detectors.) This central initial position, which will be used as the origin of the model, yields a broad coverage of the room features. If four more sets of data are then acquired, one from near each corner of the room, more detailed information about the different areas of the room, with a lessened chance of occlusion, will be obtained. The five sets of data should bring the false negative rate below 10% for subsequent recognition (Section 8.2). Recognition for these first sets of data used to build the initial model can be performed reliably since the robot is known to be in the same room: good estimates of the different positions of the robot can be obtained from the onboard odometry and used in place of the initial recognition alignments described in Section 5.1.

10.2 The "Almost Recognized" Problem

In the unlikely event that two models pass the recognition threshold test (this never occurred during testing), the robot can be moved to a different position in the same room and a second set of data can be obtained. This second set of data is combined with the first, then recognition is performed with the combined data against the competing models. Due to the greater amount of detail, the improved data set should remove any ambiguities. (More sets of data can also be obtained, if needed.) This same method is also used if no model passes the recognition test, but one or more are very close to passing the thresholds.

Before collecting the second set of data, the competing models can be compared to find where they differ the most. The robot can then be moved near this region so that the most accurate data obtained will be for this distinguishing region of the room. The robot can also be moved to search out missing data that would confirm or deny recognition that almost passed the threshold tests.

10.3 Overcoming Lighting Deficiencies and Occlusions

The robot can be moved to overcome image acquisition problems that the stereo system might be experiencing, as evidenced by deficiencies in the data obtained. For example, the scene lighting might be bad in an area of the room due to shadowing from the robot or specularities on room objects, leading to gaps in the data or obviously extraneous data points. These effects can be overcome by changing the position of the robot in the room. Likewise, occlusions that are preventing the robot from seeing some of the room features can be eliminated by moving the robot and taking another set of images. These options are not available to a fixed-position vision system.

10.4 Database Compaction

As discussed in Section 8.2, two models will occasionally be developed for the same location due to a false negative recognition. If the initial room models are built from several data sets and then checked against the existing models before adding them to the database, however, these improved initial models will often be sufficiently detailed to be recognized as corresponding to a known model. The database of models can then be compacted by combining these two models into one, with the corresponding parts of the topological world model combined (Chapter 9).

If, in spite of the detailed initial models, extraneous models are entered into the database, they can be removed later by a "cleanup" background process. When the robot is moving, its recognition subsystem (MARVEL) will be idle. During this idle time, the cleanup process can be checking for matches of model against model. Since the models will be improved over time as they are updated, the extraneous models will eventually have enough detail to be recognized as corresponding to the same location. This condition can also be checked explicitly if two different models pass the threshold tests during recognition of a data set, as mentioned in Section 10.2.

Finally, the mobile robot's topological world map and its ability to explore the world can be used together to find models that correspond to the same location. As mentioned in Section 9.4.1, by moving from its current location and finding neighboring locations, the robot can build up a small map of the local area. This local map can be matched against the world map to determine the robot's location, with additional information added to the local map through exploration until a unique match is determined.

10.5 The Computational Expense of Location Recognition

Location recognition is needed by the robot for three purposes: location verification (normal robot operation), topological map compaction, and location determination (when the robot is "lost"). The amount of processing required for the location recognition task associated with each of these purposes is different. As it turns out, the more computationally expensive location recognition occurs at times when the robot is able to devote correspondingly larger amounts of processing time to the recognition process.

Quick location recognition is required when the robot is trying to determine if it has reached an intermediate or goal location. In this case the robot is aided by the constraint that it knows where it expects to be. Thus, the search over topological map locations that might match the robot's current location is limited to just a few candidates and the search is correspondingly fast.

Location recognition is also needed to realize that two places in the topological map are the same location in order to compact the map. This identification essentially requires matching each newly-discovered location with all previously-known locations. Fortunately, the result of this identification is not required immediately after encountering a new location. The compaction of the topological map can take place as the robot moves about the world. This compaction can effectively run as a background process, with processing time devoted to the compaction when there are no more pressing needs.

Finally, location recognition is occasionally needed to determine the absolute identity of the robot's location. This need can arise if the robot becomes lost or if it is shut off and carried to a new location in its world. The first step would be, of course, to compare the location features from the current scene with the features of all the known locations in the topological map. The currently viewed features might not, however, uniquely identify a single known location. In this case, the robot would explore the world around its current location, building up a second topological map for use in identifying that location. Although the subgraph isomorphism problem is NP-complete [Garey and Johnson 1979], determination of the robot's position can be based on a matching of both the locations and the structures of the two topological maps. It will not take much exploration to build up a new topological map that uniquely matches the main world map both through isomorphism of the map structures and through consistent location recognition. Also, as more of the new topological map is built up and matched against the stored topological map, the search for matching locations is constrained by the valid matches of the two map structures. Since the robot is lost and cannot do any useful work until it has located itself with respect to its internal map, the time required to explore the local world and perform the map and location matching is well spent.

10.6 Summary

This chapter discussed some of the strategies that can be used by MARVEL to take advantage of the fact that the sensing platform it uses is a mobile robot. Most of these strategies rely on the fact that multiple sets of data for a single location can be obtained from different positions in that location. Reliable initial models can be built from several sets of data taken from the same room. Competition between models for recognition with a single set of data can be settled with additional data for that location. Likewise, recognition that almost succeeds can be verified or rejected based on information from other positions in the same room. Input problems that are due to the specific position of the robot can be overcome by simply moving the robot. Local explorations allow the robot to compare its surroundings to the world topological map to achieve data compaction or recovery from a totally "lost" state. Finally, the computational expense of these various strategies is in agreement with the time available to accomplish them. The next chapter summarizes the work that has been presented on MARVEL.

Chapter 11

Summary and Conclusions

The research described in this thesis was undertaken to answer the question, "How can an autonomous mobile robot determine where it is, based on data it has collected and models it has built, and do this in spite of a changing world and sensor error?" This question led to several areas of inquiry, each addressed in this thesis. First, how can useful information about the world be obtained to enable a mobile robot to determine its location? Second, how can this information be used to construct models of locations that can be used for recognition? Third, how can recognition be achieved so that the robot can determine its location in the world and its position in that location? Fourth, how can the models be maintained over time to reflect new information about the world and to eliminate errors that may occur? Finally, how can the mobile platform that the robot provides be used to aid in the recognition task?

11.1 Summary

The solution presented to the general question of location recognition is an implemented system named MARVEL. Through designing, building, and testing MARVEL, each of the subsidiary questions above was also addressed and answered.

An examination of the input requirements for the location recognition task led to the choice of a stereo vision input system. The stereo data provides a sparse set of simple edge features whose distribution characterizes the location from which they are derived. Roughly vertical edges are extracted from the stereo data since they provide the most accurate 3-D information available in the input. Large features from this set are preferred because they tend to correspond to large objects with stable positions. This set of simple features precludes an undesirable dependence on any single feature and allows relatively simple processing to find the location features.

The 2-D projection of the stereo features was found to retain sufficient information about their distribution for location recognition to succeed. This information is represented in an occupancy grid data abstraction that is used both as recognition data and as a basis for constructing location models. The relative simplicity of this data abstraction aids in the autonomous construction of the location models required for recognition. The models are also grid-based, with each grid point labeled with a weight that reflects its importance to the recognition process.

Recognition is achieved by comparing the current input data to existing models. Initial candidate alignments of model and data are found by considering matches of linear groupings of model and data points. A least-squares minimization technique is then used to achieve the best model/data match possible for each initial alignment. These candidate matches are compared and the best transformation of model to data is chosen for each candidate model. Using independent tests, the model that matches the data is then selected. If no model matches the data, a new location model is created from it.

If recognition succeeds, the current data and the recognition transform are used to update the matched model. These updated models provide two distinct advantages over static models. First, the models change over time to reflect the current state of the locations they represent. Second, the updates remove error that may have entered the models through the autonomous model building process.

The mobility of the robot can be used to make the recognition task easier. Since the estimated location of the robot is known, the recognition process need only consider that location and other nearby locations. The mobility of the robot also allows several data sets to be combined into the initial models, thus making them more reliable. Finally, the ability to move the robot to obtain a different view of the current location provides a way of dealing with problems such as obstructions and lighting deficiencies.

The recognition system described in this thesis was implemented and tested in real-world locations. The testing demonstrates that locations can be recognized and models can be built and maintained by an autonomous system. The real-world testing also ensured that the difficult problems were not abstracted away through the use of oversimplified simulation data.

MARVEL provides a location recognition system that fits naturally into a larger world modeling framework. The topological map world model was described along with the part that MARVEL would play in it. The topological map avoids the problems associated with cumulative error by representing the world as locations and their connections instead of relying on exact metrical information.

11.2 Comparison to Previous Work

The support that MARVEL provides for mobile robot navigation can be compared to the research done by others in the field that was mentioned in Section 1.2. This comparison shows the advances that MARVEL provides over previous work.

The system that [Drumheller 1987] built used sonar to localize the position of a robot within a known room to within ±1 foot and ±5°. MARVEL uses stereo vision to accomplish the localization to within ±1 foot and ±10°, but also recognizes the robot's location. [Faugeras *et al.* 1986] used stereo to localize the position of a robot, but this localization was accomplished with respect to the immediately preceding views of the room that the robot obtained. MARVEL, on the other hand, localizes the robot with respect to a room model instead of by tracking room features as the robot moves.

The system of [Kadanoff 1987] used infrared beacons to achieve recognition, while MARVEL does not require any changes to the environment. Also, MARVEL uses simple recognition primitives that do not require specific domain knowledge, in contrast to the work of [Nasr *et al.* 1987] that depended on high-level domain-specific landmarks. Finally, recognition by the Hilare mobile robot [Laumond 1984, 1986] was achieved by tracking the robot's movement with respect to a world model (and must therefore take odometry errors into account), whereas MARVEL performs recognition against location models and only uses the world model to select the candidate locations for recognition.

[Kuipers 1977] and [Kuipers and Byun 1987, 1988] described a world modeling system similar to the one presented in Chapter 9. Neither the mapping nor the location recognition described in these works has yet been tested on real data, whereas MARVEL's recognition, model building, and model maintenance capabilities have been tested in various locations in our laboratory. The world modeling system based on stereo data that [Faugeras *et al.* 1986] described did not provide a way of detecting or removing errors that would be incorporated into the models due to erroneous data (see [Braunegg 1989b]). Also, at each step the system proceeded incrementally from the last view of the world and did not address the question of how to use stored data from a scene that was encountered some time ago. MARVEL specifically adjusts its recognition models to detect and remove errors and achieves recognition based on models that were constructed at various times over the course of the robot's travels. MARVEL also updates these models based on currently available information to allow for changes in the visible world, while [Faugeras *et al.* 1986] seemed to assume a static world.

The work of [Sarachik 1989] successfully determined that the robot was in a rectangular room. The recognition of that room, however, depended principally on the topological structure of the world model since the room dimensions could not be determined reliably. MARVEL provides robust room recognition and position localization within the room that is independent of the world model, which is only used to limit the location model search space. MARVEL also does not depend on the rooms being rectangular, does not require a salient wall/ceiling junction or constant ceiling height, and does have possible extensions to non-office environments.

[Mataric 1990] presented a system that modeled an office environment by following and representing its walls. Individual walls could be identified, but the room itself could not be recognized. This recognition is one of the main purposes of MARVEL. Localization could be accomplished only up to determining which wall the robot was beside (with the robot constrained to be next to a wall), whereas MARVEL can determine the position and orientation of the robot anywhere in the room. The graph-based world model used was more complex that MARVEL's since the nodes of the graph represented the walls of the rooms instead of just the rooms themselves.

To support navigation, MARVEL recognizes world locations and does so using models that it builds and updates autonomously. The system also addresses some of the deficiencies in previous work in this area. MARVEL as a whole, as well as through its constituent parts. contributes to the state of the art in the field of mobile robot navigation.

11.3 Contributions and Conclusions

The main contribution of this thesis is the large, working system that was built from several different pieces to support mobile robot navigation successfully through location recognition. Not only did the system have to accomplish this goal, but it had to do so autonomously, a requirement for any exploratory mobile robot and a desirable quality for mobile robots in general. By addressing an interesting problem, this thesis takes the areas of stereo vision, model building, model maintenance, recognition, and sensing strategies and pulls them together into a coherent whole. In addition to satisfying the goal of autonomous location recognition, the thesis is itself a study in how disparate areas of expertise are brought together to form a coherent, robust system.

Starting with a well-known algorithm, a complete stereo vision input system was designed and built for MARVEL. Not only were improvements made to the stereo algorithm itself, but practical aspects in the use of stereo vision had to be addressed, such as camera and stereo calibration, selection of stereo geometry parameters (e.g., camera separation), selection of camera parameters (e.g., the lens focal lengths, balancing the inherent tradeoff between field of view and available image detail), and registration of successive stereo pairs at different rotations. All of these considerations had to be brought together to build a robust vision system that could supply input data to the larger model building and system.

A representation for the room data had to be designed that would support recognition, allow the autonomous construction and updating of models, and be derivable from the input data by the system itself. The representation choice was crucial since it was the thread that tied the various pieces of the system together. Once the specific representation was chosen, the parameter values for this data abstraction had to be selected to balance effectively the competing concerns of simplicity, detail, and computability.

The design of MARVEL's recognition algorithm had to consider not only performance issues, but also the requirements of the algorithm on the data and models needed to drive it. The data had to be readily derivable from the stereo input system. The search parameters had to be selected keeping in mind the precision of the data and model representations. Also, the algorithm in general had to reflect the philosophy of MARVEL: that it is the aggregate of the model and data features that is important for achieving recognition.

The most important contribution of this thesis is the autonomous building and maintenance of models for recognition. This is interesting because many recognition systems rely on handcrafted models that are assumed to be perfect. This thesis provides a method for autonomous model building and explicitly assumes that the models will contain errors characterized by both missing and extraneous data. Autonomous model maintenance over time is important because it provides a way of dealing with error in the models as well as responding to changes in the locations that are modeled. Without model maintenance, only static entities (locations in this thesis, objects in general) can be modeled and recognized.

The model building and maintenance ideas implemented, however, had to fit within the general framework of MARVEL. Recognition had to support the self-improving models, while the resulting updated models had to support further recognition. Both of these requirements had to be met within the framework of a representation that was derivable from the input data.

Finally, recognition and model building were bound together with the input system through the development of strategies that took advantage of the mobile robot platform of MARVEL. Deficiencies in the data, in the models, or in recognition could drive the search for new input to support these tasks. This interdependency, expressed through various sensing, recognition, and model-building strategies, points out the prime consideration in the construction of each piece of the overall system: no part of MARVEL is independent of the other parts. The system as a whole had to be considered in the design, implementation, and testing of each part of MARVEL to accomplish the primary goal, the autonomous recognition of world locations.

In conclusion, this thesis has presented MARVEL, an implemented system that supports mobile robot navigation by recognizing locations. The system is designed to operate autonomously with no specific *a priori* world knowledge. It uses stereo vision to obtain information about the world and builds data representations from this information. The data representations are in turn used to build models of the world locations that are encountered. The models are compared with newly acquired input data to recognize the robot's current location. The model for this location is then updated based on the recognition and the new data. These location recognition, model building, and model maintenance capabilities that MARVEL provides fit naturally into a larger world modeling system that is under development for use with an autonomous mobile robot. Various strategies that take advantage of the mobility of the robot and the information in the world model are used by MARVEL for achieving recognition. All of these pieces combine to make MARVEL a complete, tested system that supports navigation by a mobile robot by recognizing world locations.

Chapter 12

Further Work

MARVEL is a complete location recognition system that has been designed, implemented, and tested. As with all interesting work, however, research into one question leads to others, solutions to problems suggest different paths for solving them, and the knowledge gained through designing, building, and testing a system can always be extended. To finish the discussion of the location recognition system MARVEL, which is the focus of this thesis, several areas of further work are presented below.

12.1 Recognition Algorithm Improvements

The recognition algorithm described in Chapter 5 uses linear groupings of data. Other methods of grouping can also be used to determine the initial alignments of model and data [Jacobs 1988]. The model and data points might be associated based on other structures, such as curves, or by proximity. Properties of these structures or groupings could then be used to determine initial alignments of model and data.

The initial alignments are used to overcome the problem of local minima that occurs in the least-squares minimization process for recognition. Methods such as simulated annealing could be explored for finding the correct solution to the minimization independently of the initial match.

The data MARVEL uses for recognition is based on vertical stereo features, with all sufficiently long verticals in the floor-to-ceiling range treated equally. Height information associated with these verticals could be preserved for use in recognition. One way of accomplishing this is to form several representations for each set of data, based on the height ranges of the verticals that were seen. For example, three representations, based on features found in low, middle, and high height ranges could be used. With this scheme, data features lying near the floor would not be matched against model features found near the ceiling. The sizes of these overlapping height ranges would determine how close two features should be in the vertical (z) direction to be considered as candidates for matching. This height information could also be used to determine where occlusions are expected. For example, low features would be expected to occlude other low features, but not high features. The low features might be furniture in a room and the high features paintings on the walls. These expected occlusions could be taken into account when determining what parts of the models and how much of the models must be matched in order to achieve recognition.

12.2 Model Building and Maintenance Improvements

The weights of a model's points are determined by their use in recognizing that model. The weights could also be adjusted based on the importance of individual points to distinguishing one model from another. This might lead to a set of weights for the points of a model, with the particular weights used depending on which other models were being considered for recognition.

The age of a model point, which affects how its weight is increased or decreased, is based on a count of the data sets in which it was seen. A true age based on an internal clock could also be used in this regard. A model that has not been used in a long while would then be suspect (its point weights lowered) due to the assumption that much could have changed in the long time since the model was last verified.

Both a linear and an exponential model point weight update function were discussed in Chapter 7. The use of other functions is also possible. Different methods of retracting model points could also be explored. One obvious candidate is the complete retraction of a model point if it is not seen after a suitable hysteresis delay.

12.3 Real-Time Performance

MARVEL was designed, implemented, and tested to explore the issues of world location recognition, model building, and model maintenance. The implementation focused on proof of concept rather than practicality for a mobile robot. Some possibilities exist, however, for achieving "real-time" performance of MARVEL by speeding up the stereo processing and the least-squares minimization of the recognition module.

The 3-D stereo data is projected to two dimensions before being used for model building and recognition. By using cylindrical lenses and single-scanline cameras, the compression to 2-D can be accomplished optically and a single-scanline stereo algorithm performed. The Mobile Robot Group at the MIT Artificial Intelligence Laboratory has implemented such a stereo system that operates at higher than television frame rates, which is the usual standard for real-time operation [Viola 1990]. If this method is used, however, room features other than long, vertical edges will be matched. Therefore, the utility of the features detectable in these compressed images for location recognition must be investigated.

The least-squares minimization process consists of a large set of array operations. These can be modeled as vector operations and implemented on Digital Signal Processing (DSP) integrated circuit chips for fast processing.

12.4 Integration with Other Systems

A mobile robot will be called on to perform many tasks in its environment. The robot can be looked upon as a mobile platform that transports the systems that execute these tasks. By determining the robot's location and its position and orientation within that location, MARVEL allows the robot to bring these systems to bear where they are needed. Specific information about the positions in the locations that are important for these tasks can be included in MARVEL's location models. It is also possible that some of the information gathered by MARVEL can be directly used to help perform these tasks. For instance, MARVEL obtains 2-D images of and 3-D stereo information about its surroundings. This information can also be used as input to the task systems. The position of the robot, as determined by MARVEL, provides important information for a system designed to find certain objects (e.g., intruders) in the robot's world. For example, the position information can be used by this search system to determine its next course of action. The whole area of interaction between MARVEL and other onboard systems is open for future research, with many possibilities provided by these other systems and the tasks they perform.

12.5 Input from Other Sensors

MARVEL was tested using input from a stereo vision system. Other sensors could also provide well-localized information about features in the world that could be used for recognition. Obvious candidates for use with a mobile robot include sonar and laser ranging. The power requirements and obtrusive nature of these active sensing systems would have to be taken into account when designing the robot, of course. Information from different sensors could also be combined. For instance, stereo's good angular resolution and variable depth resolution is complemented well by the good depth resolution and wide angular resolution of sonar data.

12.6 Extensions to Other Environments

MARVEL was discussed in the context of an indoor, office-type environment. Work needs to be done to extend MARVEL to other domains. One domain of interest would be outdoor areas. MARVEL's use of isolated features for recognition is amenable to such an extension through the redefinition of the features. Objects such as barns, houses, and trees could be detected and used by MARVEL. As with the vertical stereo edges, the distribution of these features about the robot would supply the information necessary to recognize its location.

12.7 Summary

This chapter discussed several possible extensions to MARVEL. Improvements to the recognition algorithm were suggested, centering around the use of other groupings to obtain the initial model/data alignments. Different methods of building and maintaining the models were mentioned, based on other methods of incorporating new data into existing models. Some hardware speedups of the stereo vision and recognition modules of MARVEL are possible. Information from sonar and other sensors could be incorporated into the location models. Finally, ways of extending MARVEL to other environments should be investigated, based on the use of sparse yet robust features in the locations to be recognized.

Although any of the extensions mentioned above could be explored further, as described in this thesis MARVEL is a complete, working system. The stereo input, model building, recognition, and model maintenance facets of MARVEL have all been implemented. The testing of these various pieces and the system as a whole demonstrate that MARVEL indeed supports mobile robot navigation through recognizing world locations.

Appendix A

Least-Squares Transform Refinement

Section 5.2 outlined the least-squares minimization method used to refine a set of model/data initial alignments. The mathematics for this least-squares minimization is presented in this chapter.

Given an initial transformation $\mathbf{t}_0 = (x_0, y_0, \theta_0)$, a set of model points M, and a set of data points D, and a set of search radii $\{r_i\}$, select the largest search radius $r = r_0$. Transform each model point:

$$\hat{\mathbf{m}}_i = \begin{bmatrix} \hat{m}_{ix} \\ \hat{m}_{iy} \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} m_{ix} \\ m_{iy} \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \qquad \forall \mathbf{m}_i \in M.$$

For these transformed model points, find all possible model/data pairs:

$\forall \hat{\mathbf{m}}_i$, choose $\mathbf{d}_i \in D$ such that

- $\|\hat{\mathbf{m}}_i, \mathbf{d}_i\|$ is minimized
- $\|\hat{\mathbf{m}}_i, \mathbf{d}_i\| \le r$

(if such a $\mathbf{d}_i$ exists).

The transformed model points $\hat{\mathbf{m}}_i$ that have been matched can be expressed as a vector $\hat{\mathbf{m}}$ in terms of their x- and y-coordinates:

$$\hat{\mathbf{m}} = \begin{bmatrix} \hat{m}_{1x} \\ \hat{m}_{1y} \\ \hat{m}_{2x} \\ \hat{m}_{2y} \\ \vdots \end{bmatrix} = f_{\mathbf{m}}(\mathbf{t}_0) = f_{\mathbf{m}}(x_0, y_0, \theta_0) \tag{A.1}$$

$$= \begin{bmatrix} x_0 & + & m_{1x}\cos\theta_0 & - & m_{1y}\sin\theta_0 \\ y_0 & + & m_{1x}\sin\theta_0 & + & m_{1y}\cos\theta_0 \\ x_0 & + & m_{2x}\cos\theta_0 & - & m_{2y}\sin\theta_0 \\ y_0 & + & m_{2x}\sin\theta_0 & + & m_{2y}\cos\theta_0 \\ & & \vdots & & \end{bmatrix}.$$

Ideally, the transformed model points will exactly coincide with the data points:

$$\hat{\mathbf{m}} = \begin{bmatrix} \hat{m}_{1x} \\ \hat{m}_{1y} \\ \hat{m}_{2x} \\ \hat{m}_{2y} \\ \vdots \end{bmatrix} = \begin{bmatrix} d_{1x} \\ d_{1y} \\ d_{2x} \\ d_{2y} \\ \vdots \end{bmatrix} = \mathbf{d}.$$

Since the initial transform estimate is not exact, however, they will not coincide. We wish to obtain a new transform estimate

$$\mathbf{t}_1 = \mathbf{t}_0 + \Delta\mathbf{t} = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

that will bring $\hat{\mathbf{m}}$ and $\mathbf{d}$ closer together. (Due to noise, we will not, in general, be able to make the transformed model points coincide with the data points.) If $\hat{\mathbf{m}}$ is the set of transformed model points due to transform $\mathbf{t}_0$, then these same model points due to $\mathbf{t}_1$ will be

$$\hat{\mathbf{m}} + \Delta\hat{\mathbf{m}}$$

where $\Delta\hat{\mathbf{m}}$ is the change in the model points due to $\Delta\mathbf{t}$.

The difference between the new set of transformed model point coordinates and the data point coordinates is

$$\begin{aligned} \mathbf{e} &= (\hat{\mathbf{m}} + \Delta\hat{\mathbf{m}}) - \mathbf{d} \\ &= (\hat{\mathbf{m}} - \mathbf{d}) + \Delta\hat{\mathbf{m}}. \end{aligned} \tag{A.2}$$

Recall from Equation A.1 that

$$\hat{\mathbf{m}} = f_{\mathbf{m}}(\mathbf{t}_0) = f_{\mathbf{m}}(x_0, y_0, \theta_0).$$

For small $\Delta\hat{\mathbf{m}}$, $\Delta\hat{\mathbf{m}} \approx d\hat{\mathbf{m}}$. The chain rule then implies that

$$\begin{aligned} \Delta\hat{\mathbf{m}} &= \Delta f_{\mathbf{m}}(x_0, y_0, \theta_0) \\ &= \left.\frac{\partial f_{\mathbf{m}}}{\partial x}\right|_{\mathbf{t}_0} \Delta x + \left.\frac{\partial f_{\mathbf{m}}}{\partial y}\right|_{\mathbf{t}_0} \Delta y + \left.\frac{\partial f_{\mathbf{m}}}{\partial \theta}\right|_{\mathbf{t}_0} \Delta\theta \\ &= J_0 \Delta\mathbf{t} \end{aligned}$$

where J_0 is the Jacobian matrix

$$J_0 = \begin{bmatrix} \frac{\partial \hat{m}_{1x}}{\partial x} & \frac{\partial \hat{m}_{1x}}{\partial y} & \frac{\partial \hat{m}_{1x}}{\partial \theta} \\ \frac{\partial \hat{m}_{1y}}{\partial x} & \frac{\partial \hat{m}_{1y}}{\partial y} & \frac{\partial \hat{m}_{1y}}{\partial \theta} \\ \frac{\partial \hat{m}_{2x}}{\partial x} & \frac{\partial \hat{m}_{2x}}{\partial y} & \frac{\partial \hat{m}_{2x}}{\partial \theta} \\ \frac{\partial \hat{m}_{2y}}{\partial x} & \frac{\partial \hat{m}_{2y}}{\partial y} & \frac{\partial \hat{m}_{2y}}{\partial \theta} \\ \vdots & \vdots & \vdots \end{bmatrix}_{\mathbf{t}_0}.$$

Thus, the error term can be expressed as

$$\mathbf{e} = (\hat{\mathbf{m}} - \mathbf{d}) + J_0 \Delta\mathbf{t}.$$

To refine the transform estimate by adjusting $\Delta\mathbf{t}$, we must minimize this error. Note that an assumption from the chain rule is that for small $\Delta\theta$ about the initial guess rotation θ, $\frac{\partial f_{\mathbf{m}}}{\partial \theta}$ is approximately linear, thus letting us apply the least-squares minimization.

The weights of each model point can be used to influence the importance of matching the individual model points. The error vector **e** is

$$\begin{bmatrix} \vdots \\ e_{ix} \\ e_{iy} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \hat{m}_{ix} - d_{ix} \\ \hat{m}_{iy} - d_{iy} \\ \vdots \end{bmatrix} + \begin{bmatrix} \vdots \\ \frac{\partial \hat{m}_{ix}}{\partial x}\Delta x_0 + \frac{\partial \hat{m}_{ix}}{\partial y}\Delta y_0 + \frac{\partial \hat{m}_{ix}}{\partial \theta}\Delta \theta_0 \\ \frac{\partial \hat{m}_{iy}}{\partial x}\Delta x_0 + \frac{\partial \hat{m}_{iy}}{\partial y}\Delta y_0 + \frac{\partial \hat{m}_{iy}}{\partial \theta}\Delta \theta_0 \\ \vdots \end{bmatrix},$$

where the components e_{ix} and e_{iy} result from the match of transformed model point $\hat{m}_i$. If w_i is the weight of model point $\mathbf{m}_i$, then the squared errors associated with the coordinates of $\hat{\mathbf{m}}_i$ should be weighted. If we let

$$W = \begin{bmatrix} w_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & w_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & w_n \end{bmatrix},$$

then Equation A.2 becomes

$$\mathbf{e}^T W \mathbf{e} = [(\hat{\mathbf{m}} - \mathbf{d}) + J_0 \Delta \mathbf{t}]^T \, W \, [(\hat{\mathbf{m}} - \mathbf{d}) + J_0 \Delta \mathbf{t}],$$

the weighted squared error. Letting $\mathbf{h} = \hat{\mathbf{m}} - \mathbf{d}$, we have

$$\begin{aligned} \mathbf{e}^T W \mathbf{e} &= [\mathbf{h} + J_0 \Delta \mathbf{t}]^T \, W \, [\mathbf{h} + J_0 \Delta \mathbf{t}] \\ &= \mathbf{h}^T W \mathbf{h} - \Delta \mathbf{t}^T J_0^T W \mathbf{h} - \mathbf{h}^T W J_0 \Delta \mathbf{t} + \Delta \mathbf{t}^T J_0^T W J_0 \Delta \mathbf{t}. \end{aligned}$$

Note that for $|\hat{\mathbf{m}}| > 3$, this system of equations is overdetermined. We wish to minimize this weighted squared error with respect to $\Delta \mathbf{t}$, so taking the partial derivative and setting it equal to zero

$$\frac{\partial \mathbf{e}^T W \mathbf{e}}{\partial \Delta \mathbf{t}} = 0 - J_0^T W \mathbf{h} - J_0^T W \mathbf{h} + 2 J_0^T W J_0 \Delta \mathbf{t} = 0$$

yields

$$\Delta \mathbf{t} = [J_0^T W J_0]^{-1} J_0^T W \mathbf{h}.$$

Repeatedly using this least-squares minimization, we iterate to refine the transform

$$\mathbf{t}_{i+1} = \mathbf{t}_i + \Delta \mathbf{t}$$

until either the weighted transform variance is small enough or is changing slowly enough

$$S^2 < \delta_1 \qquad \text{or} \qquad \Delta S^2 < \delta_2.$$

The weighted transform variance is used to determine how well the current transform matches the model points to the data points. It is the weighted sample variance S^2 as determined by summing the weighted squared distances between each model point and its closest data point, then dividing by the number of model points minus one [Walpole and Myers 1978].

$$S^2 = \frac{\sum_{i=1}^{n} w_i (\mathbf{d}_i - \hat{\mathbf{m}}_i)^2}{n-1}$$

where

$$\begin{aligned} \hat{\mathbf{m}}_i &= \text{the transformed model point } \mathbf{m}_i \\ \mathbf{d}_i &= \text{the data point that } \hat{\mathbf{m}}_i \text{ matched} \\ w_i &= \text{the weight assigned to model point } \mathbf{m}_i \end{aligned}$$

This process is then repeated for successively smaller search radii r_i. The transform found at the smallest search radius is considered to be the best transform for the given initial alignment.

For a set of refined model/data transforms that were determined from a set of initial alignments, the transform selected as best matching the model to the data is the one that maximizes the percentage of model matched $\frac{||\hat{\mathbf{m}}||}{||M||}$. If two transforms match the same number of model points, the one with the smallest sample variance S^2 is chosen.

Appendix B

Model Update Algorithm

This appendix presents the algorithm used to update the models that was described in Chapter 7. The transform mapping data points to model points is assumed to correspond to correct recognition of the data. The following notation is used:

$D =$ set of transformed data points $\{d\}$

$M =$ set of current model points $\{m\}$

$t_r =$ rise time of weight update function

$h =$ weight reduction hysteresis

$a_m =$ age of model point m

$c_m =$ count of model point m

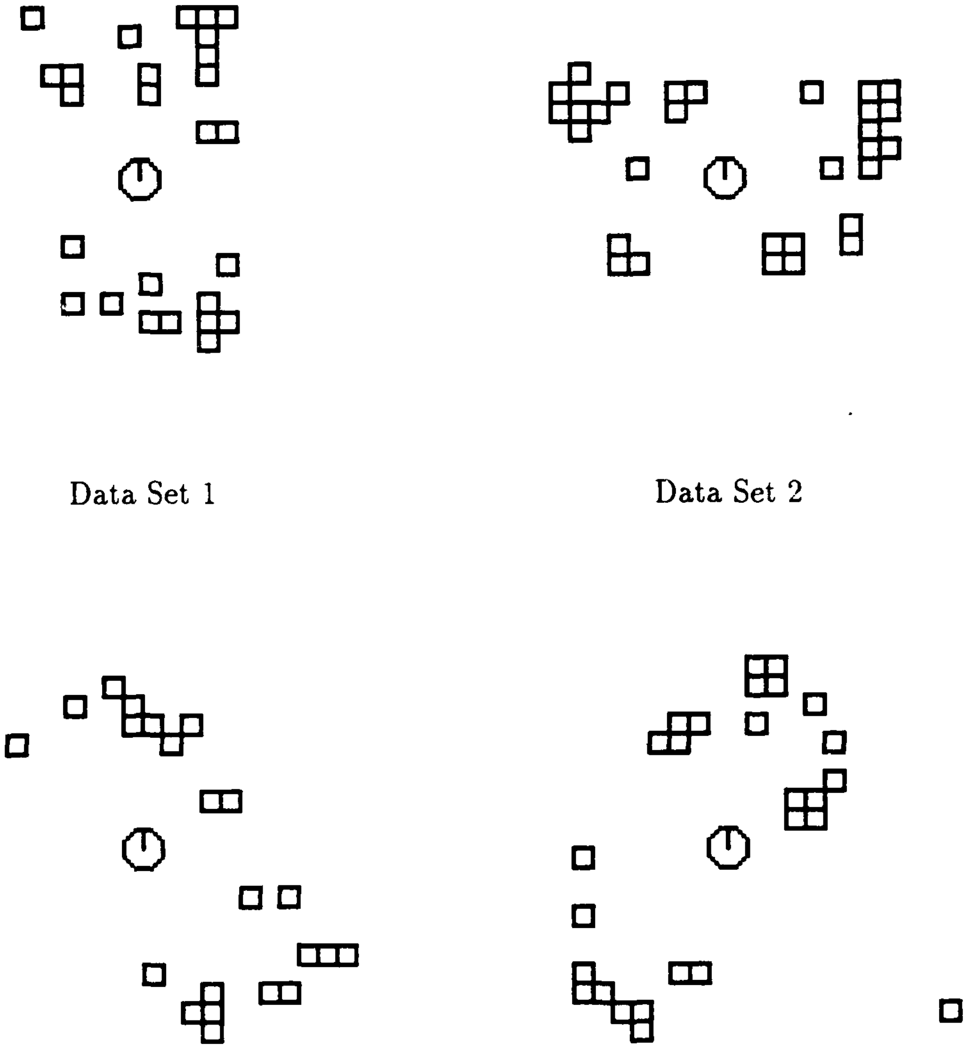
$W_m = W_m(c\,; t_r) =$ weight of model point m

B.1 Model Initialization (Data Set 1)

```
$M \leftarrow \emptyset$

 loop for $d \in D$
   $m \leftarrow d$
   $a_m \leftarrow 0$
   $c_m \leftarrow 1$
   $W_m \leftarrow W_m(1; t_r)$
   $M \leftarrow M + m$
 endloop
```

B.2 Model Update (Data Sets 2–n)

```
$M_{new} \leftarrow \emptyset$

;;; Consider each current model point in turn
loop for $m \in M$

  ;; Update model point age
  $A_m \leftarrow A_m + 1$
  $\delta_{min} \leftarrow \infty$

  ;; Find closest data point
  loop for $d \in D$
    if $\|m - d\| < \delta_{min}$
      then
        $\delta_{min} \leftarrow \|d - m\|$
        $d_{min} \leftarrow d$
    endif
  endloop

  ;; Evaluate if matched data point is close enough
  if $\delta_{min} < \epsilon$
    then
      ;; Update model point properties based on match
      $a_m \leftarrow 0$
      $c_m \leftarrow \min(t_r, c_m + 1)$
      $W_m \leftarrow W_m(c_m; t_r)$
      $M_{new} \leftarrow M_{new} + m$
      ;; Remove this data point from further consideration
      $D \leftarrow D - d$
      else
        ;; Update model point properties based on no match
        if $a_m \geq h$
          then
            ;; Point exceeds hysteresis, so fade it
            $c_m \leftarrow c_m - 1$
            if $c_m > 0$
              $W_m \leftarrow W_m(c_m; t_r)$
              $M_{new} \leftarrow M_{new} + m$
            endif
          else
            ;; Point does not exceed hysteresis, so keep it as is
            $M_{new} \leftarrow M_{new} + m$
        endif
    endif
endloop

;;; Add unmatched data points to model
loop for $d \in D$
  $m \leftarrow d$
  $a_m \leftarrow 0$
  $c_m \leftarrow 1$
  $W_m \leftarrow W_m(1; t_r)$
  $M_{new} \leftarrow M_{new} + m$
endloop

;;; Make the current model now be the updated model
$M \leftarrow M_{new}$
```



Data Set 1

Data Set 2

Data Set 3

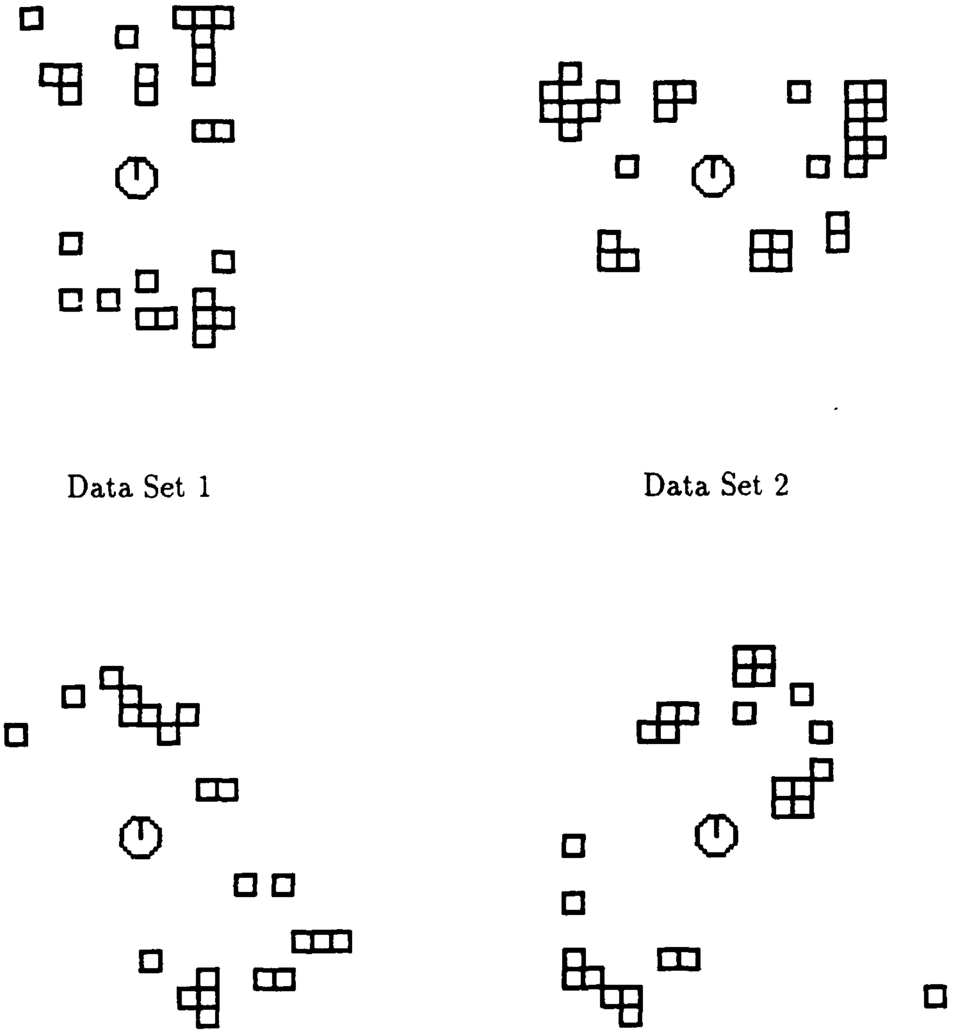
Data Set 4

Figure C-1: Room 913 data sets 1–4.

Appendix C

Data and Models for Room 913

This appendix contains data taken from Room 913 in the MIT Artificial Intelligence Laboratory and the resulting models. Model 1 was created from Data Set 1. Model 2 is the update of Model 1 based on recognition with Data Set 2. Model 3 is the update of Model 2 based on recognition with Data Set 3, and so on. The weight of a model point is indicated by its darkness. Low weight points are shown as light grey squares while darker squares denote more heavily weighted model points. The model updates used the linear weight update of Equation 7.1 with rise time $t_r = 5$ and hysteresis $h = 4$.



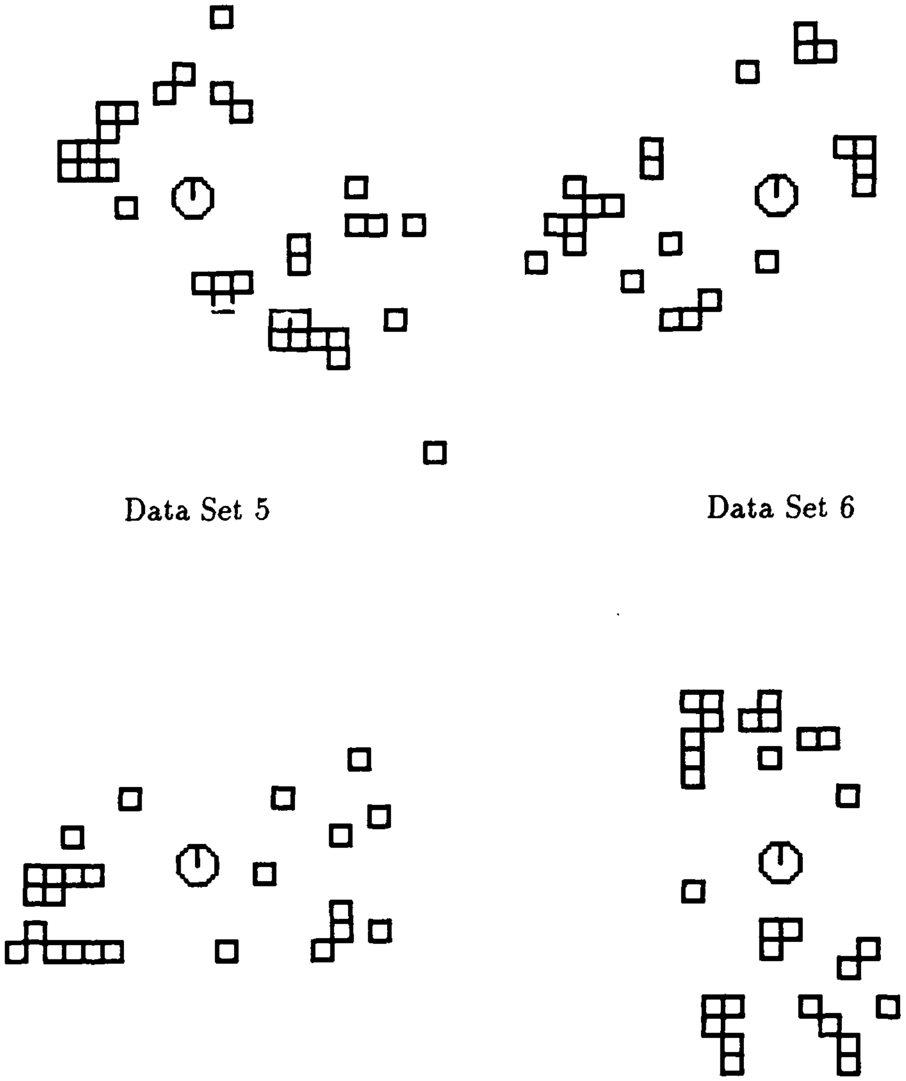
Data Set 1

Data Set 2

Data Set 3

Data Set 4

Figure C-1: Room 913 data sets 1–4.



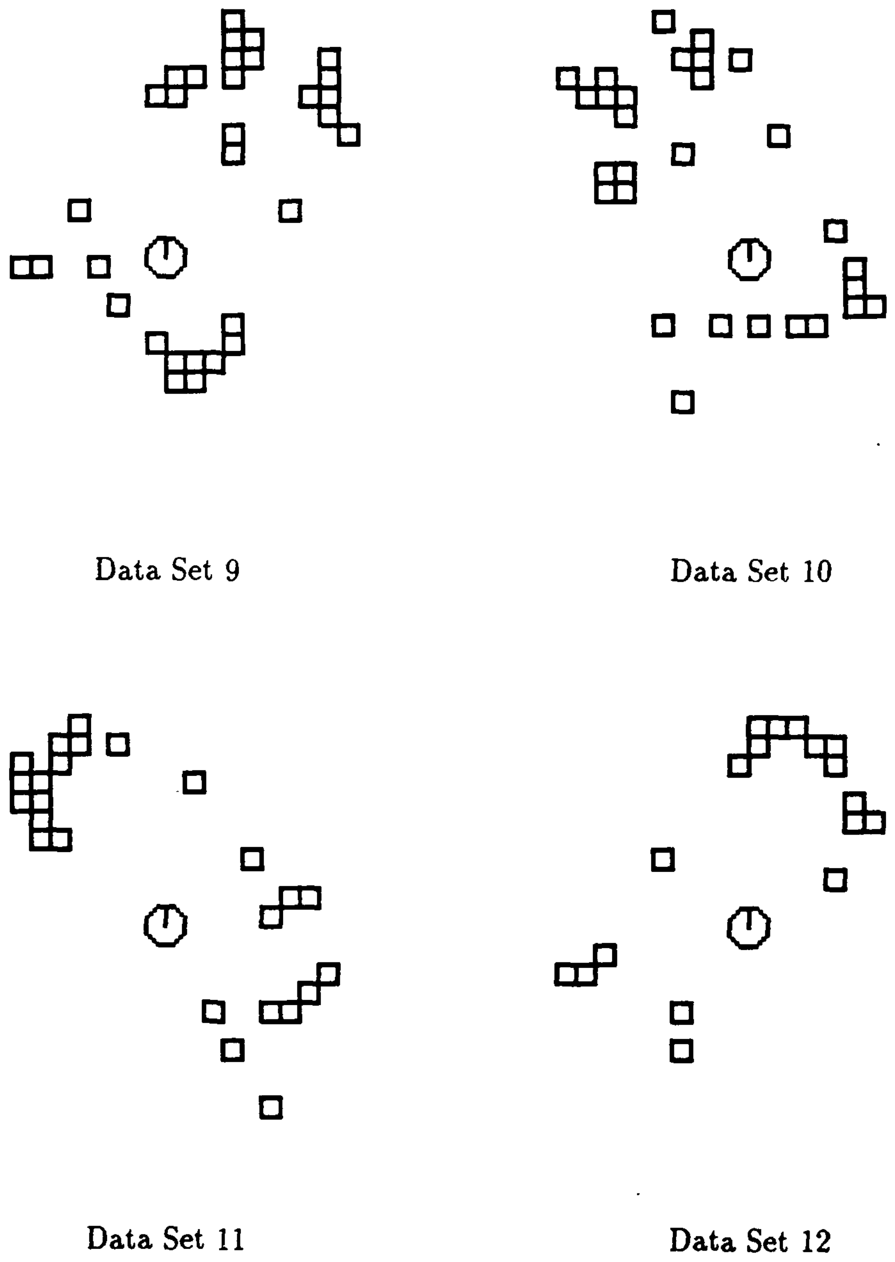
Data Set 5

Data Set 6

Data Set 7

Data Set 8

Figure C-2: Room 913 data sets 5–8.



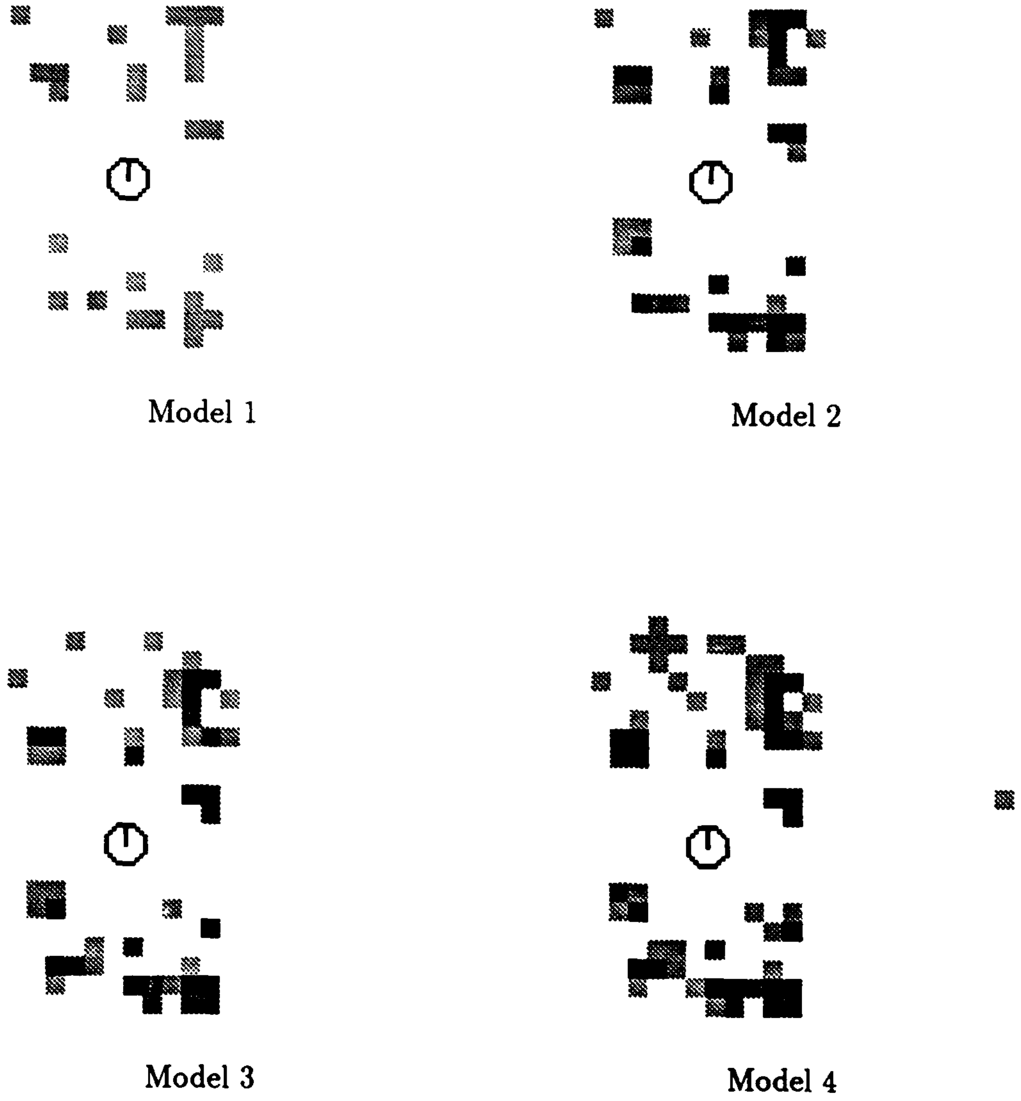
Data Set 9

Data Set 10

Data Set 11

Data Set 12

Figure C-3: Room 913 data s*ts 9–12.



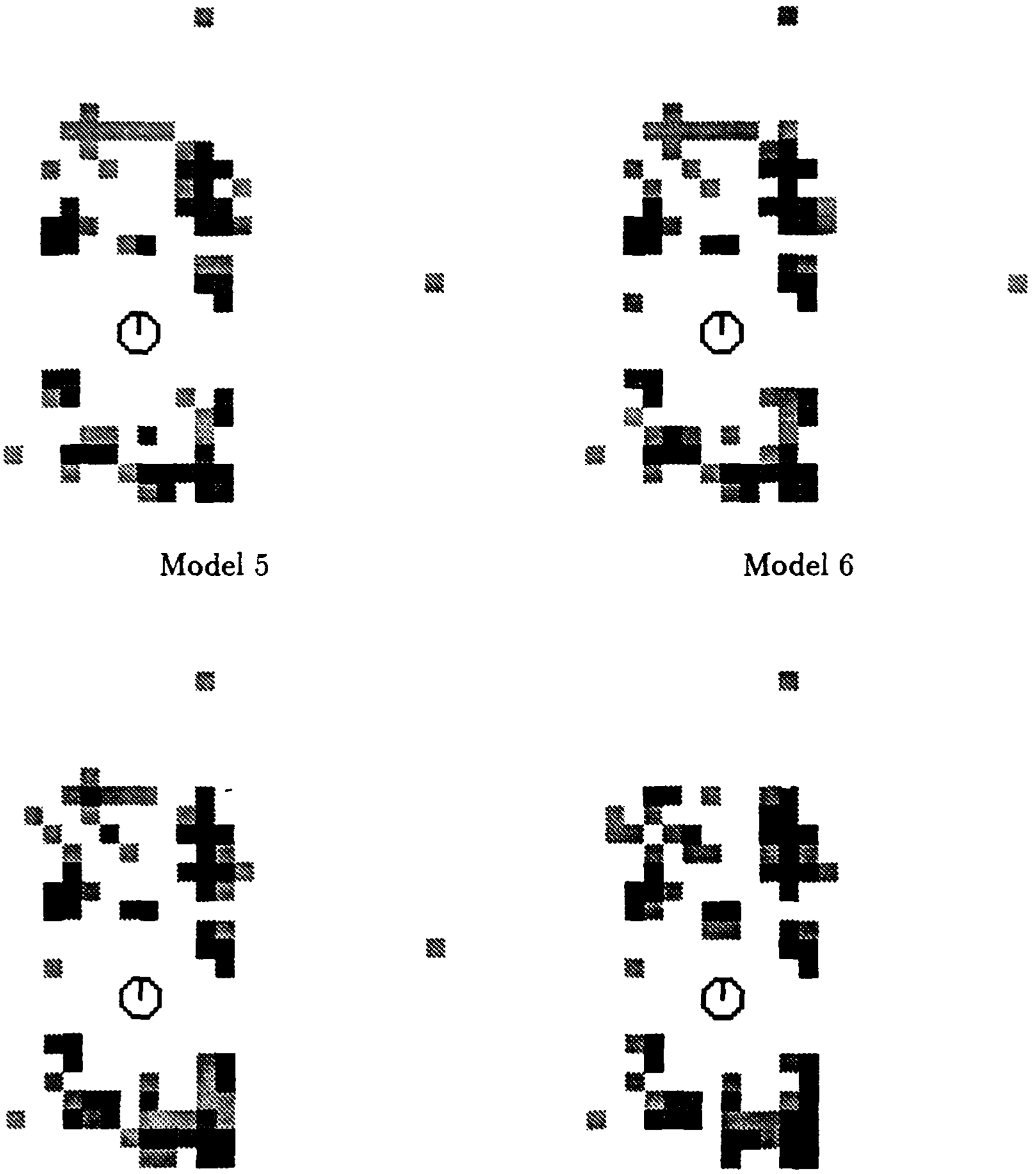
Model 1

Model 2

Model 3

Model 4

Figure C-4: Room 913 models 1–4.



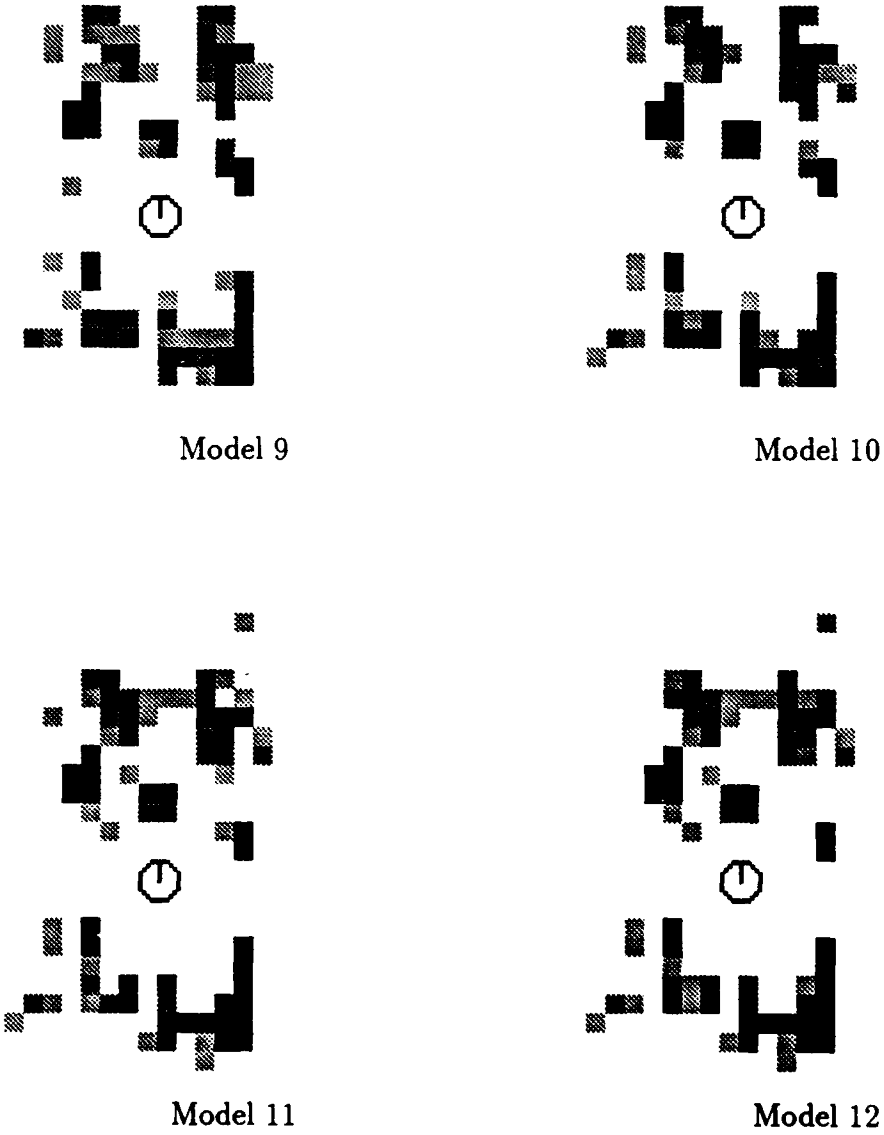
Model 5

Model 6

Model 7

Model 8

Figure C-5: Room 913 models 5–8.



Model 9

Model 10

Model 11

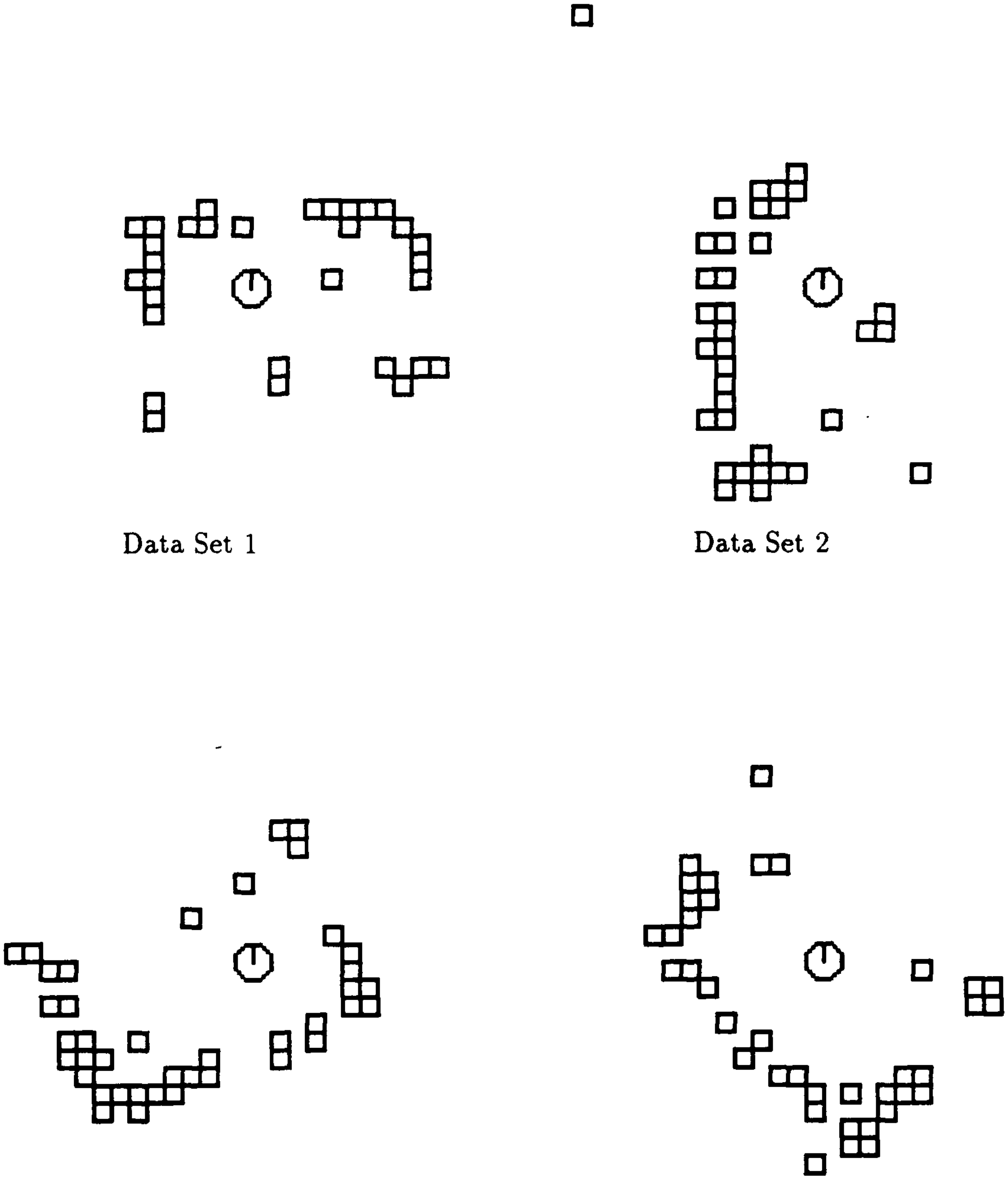
Model 12

Figure C-6: Room 913 models 9–12.

Appendix D

Data and Models for Room 914

This appendix contains data taken from Room 914 in the MIT Artificial Intelligence Laboratory and the resulting models. Model 1 was created from Data Set 1. Model 2 is the update of Model 1 based on recognition with Data Set 2. Model 3 is the update of Model 2 based on recognition with Data Set 3, and so on. The weight of a model point is indicated by its darkness. Low weight points are shown as light grey squares while darker squares denote more heavily weighted model points. The model updates used the linear weight update of Equation 7.1 with rise time $t_r = 5$ and hysteresis $h = 4$.



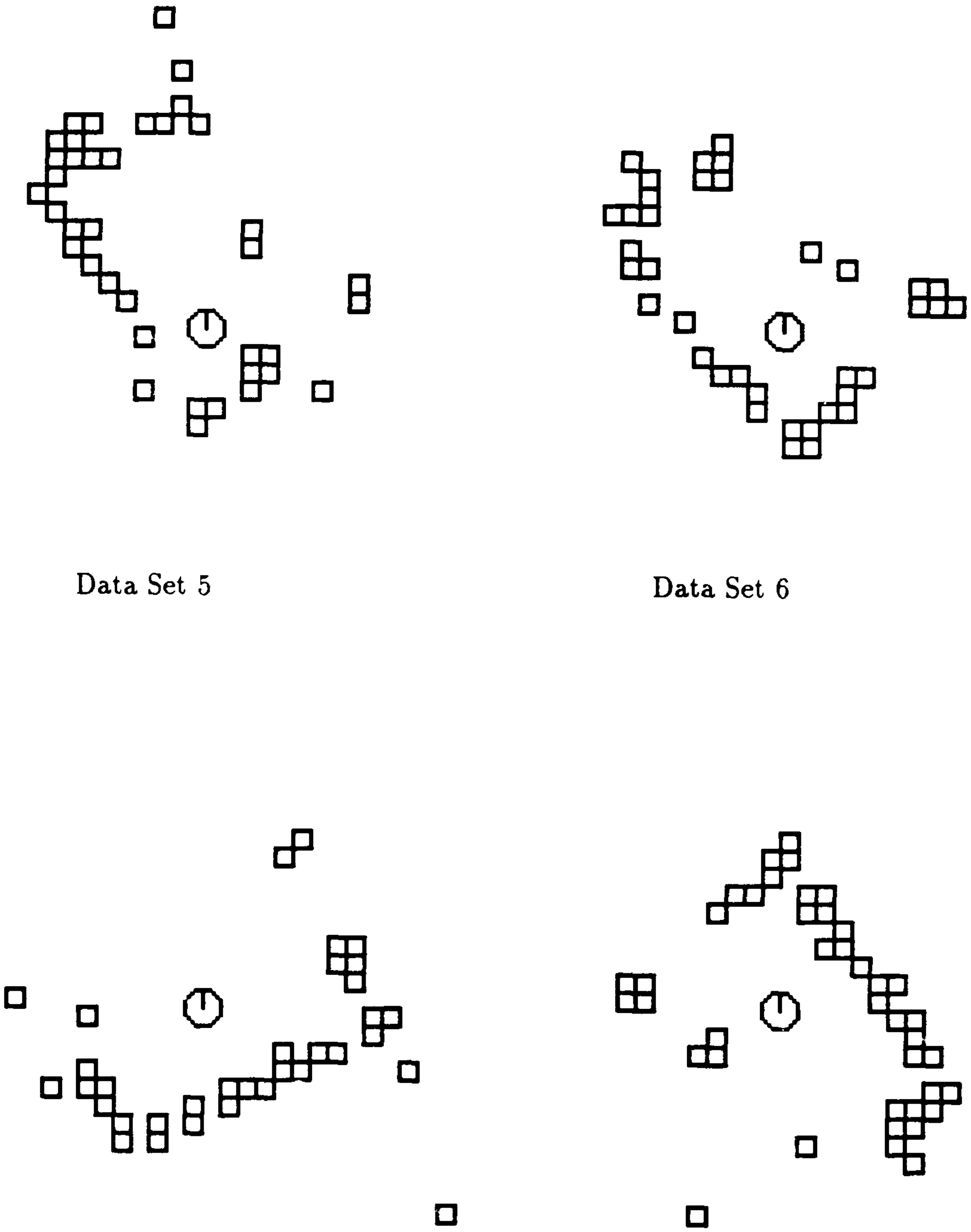
Data Set 1

Data Set 2

Data Set 3

Data Set 4

Figure D-1: Room 914 data sets 1–4.



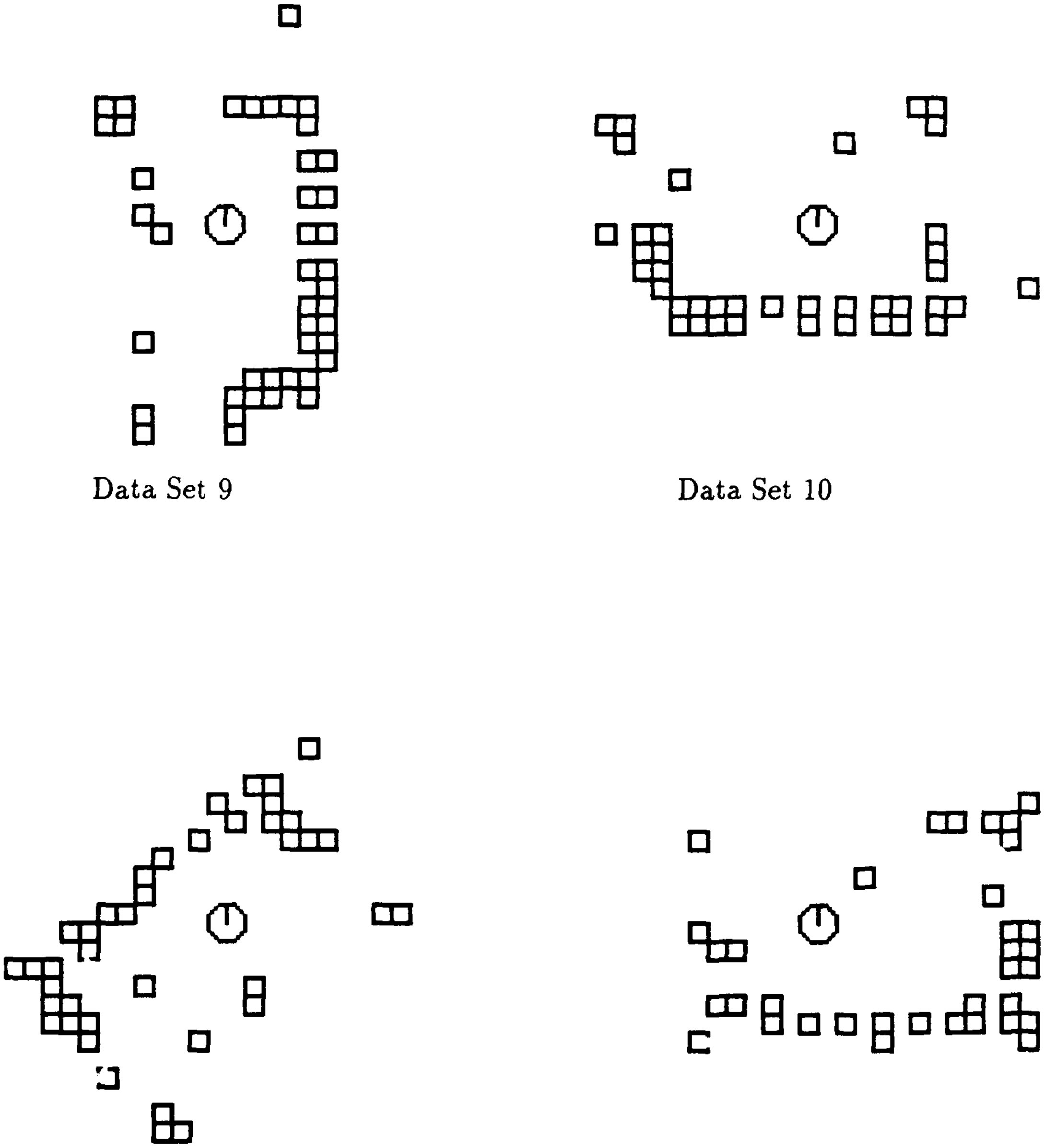
Data Set 5

Data Set 6

Data Set 7

Data Set 8

Figure D-2: Room 914 data sets 5–8.



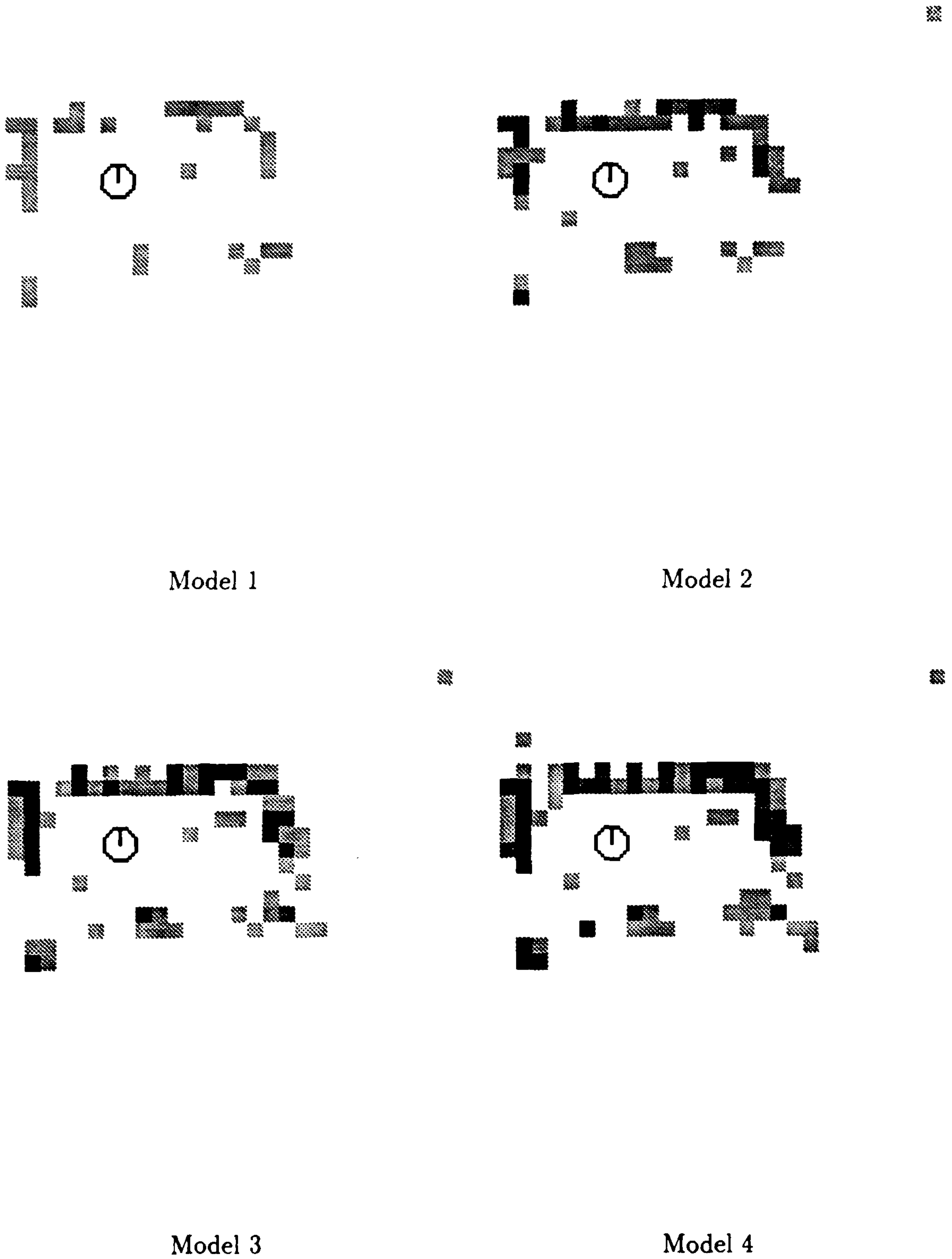
Data Set 9

Data Set 10

Data Set 11

Data Set 12

Figure D-3: Room 914 data sets 9–12.



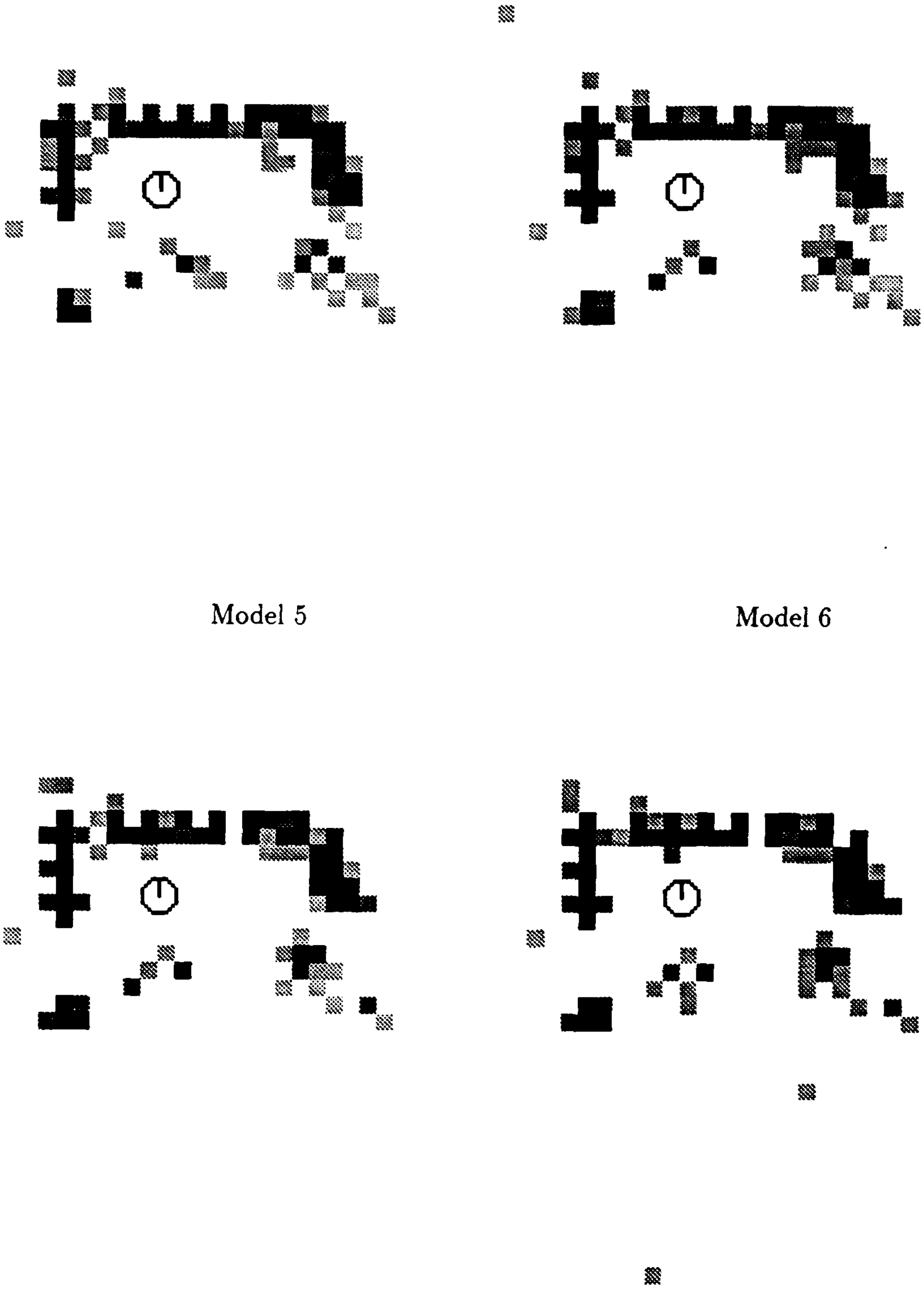
Model 1

Model 2

Model 3

Model 4

Figure D-4: Room 914 models 1–4.



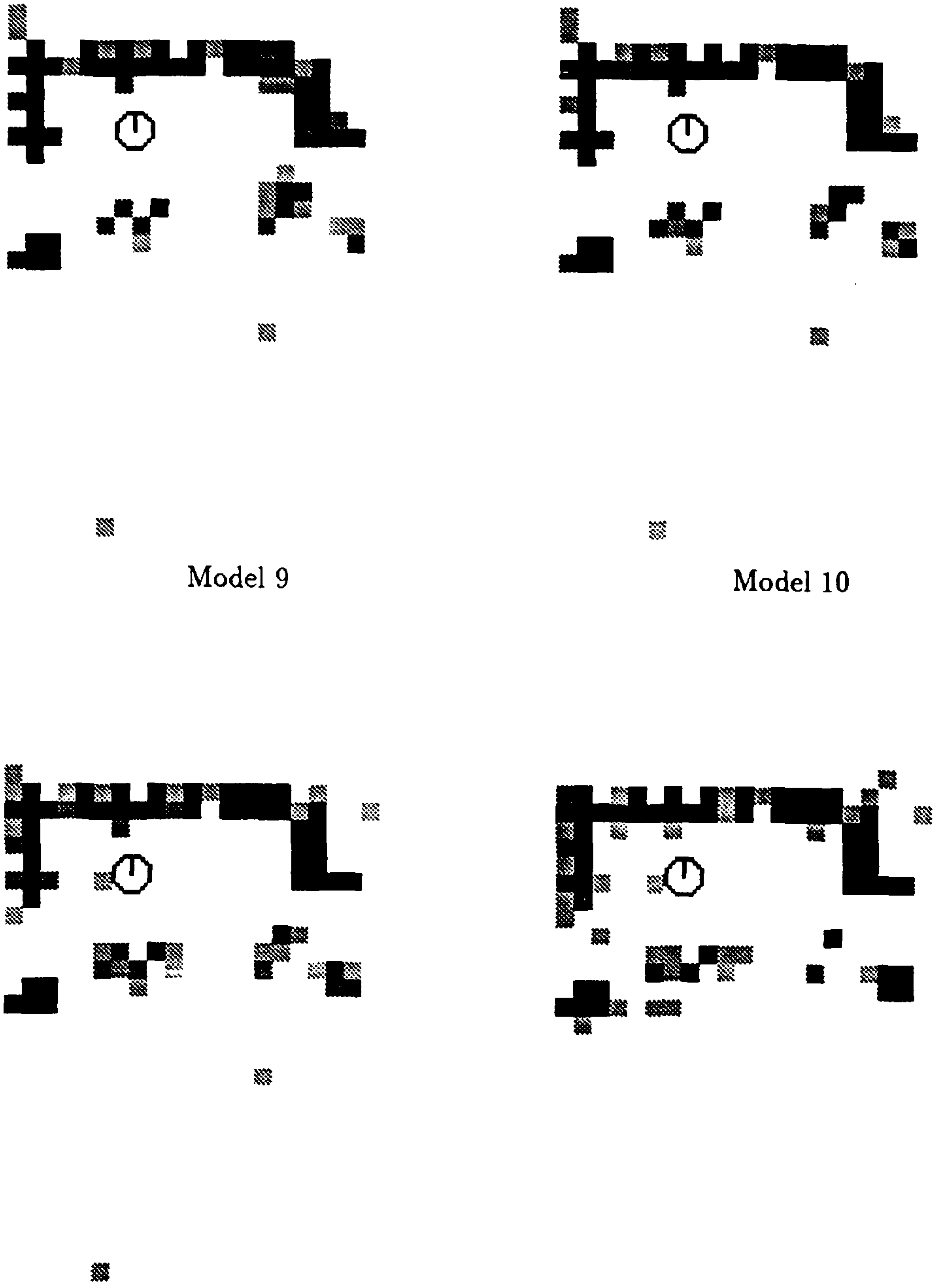
Model 5

Model 6

Model 7

Model 8

Figure D-5: Room 914 models 5–8.



Model 9

Model 10

Model 11

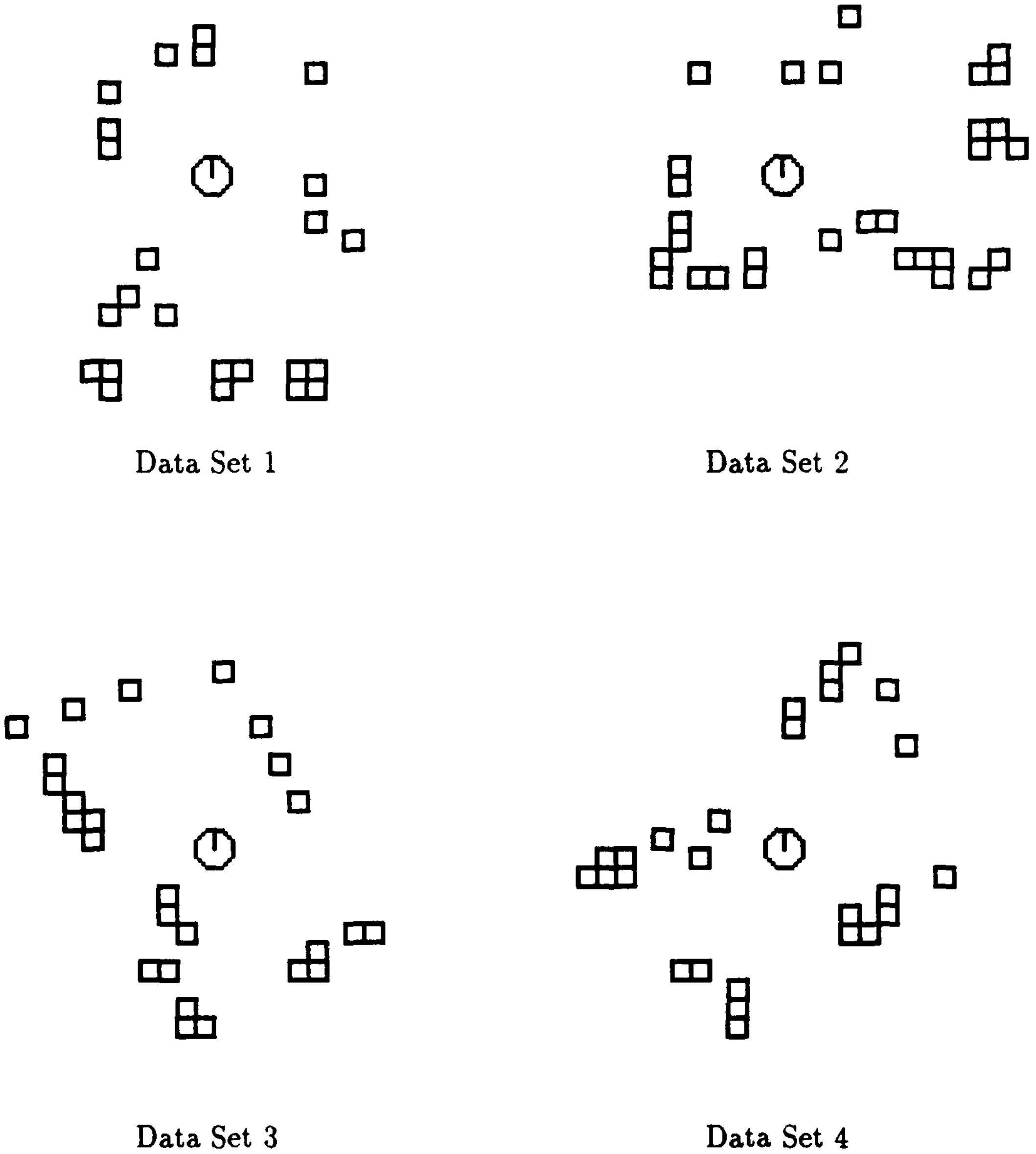
Model 12

Figure D-6: Room 914 models 9–12.

Appendix E

Data and Models for Room 915

This appendix contains data taken from Room 915 in the MIT Artificial Intelligence Laboratory and the resulting models. Model 1 was created from Data Set 1. Model 2 is the update of Model 1 based on recognition with Data Set 2. Model 3 is the update of Model 2 based on recognition with Data Set 3, and so on. The weight of a model point is indicated by its darkness. Low weight points are shown as light grey squares while darker squares denote more heavily weighted model points. The model updates used the linear weight update of Equation 7.1 with rise time $t_r = 5$ and hysteresis $h = 4$.



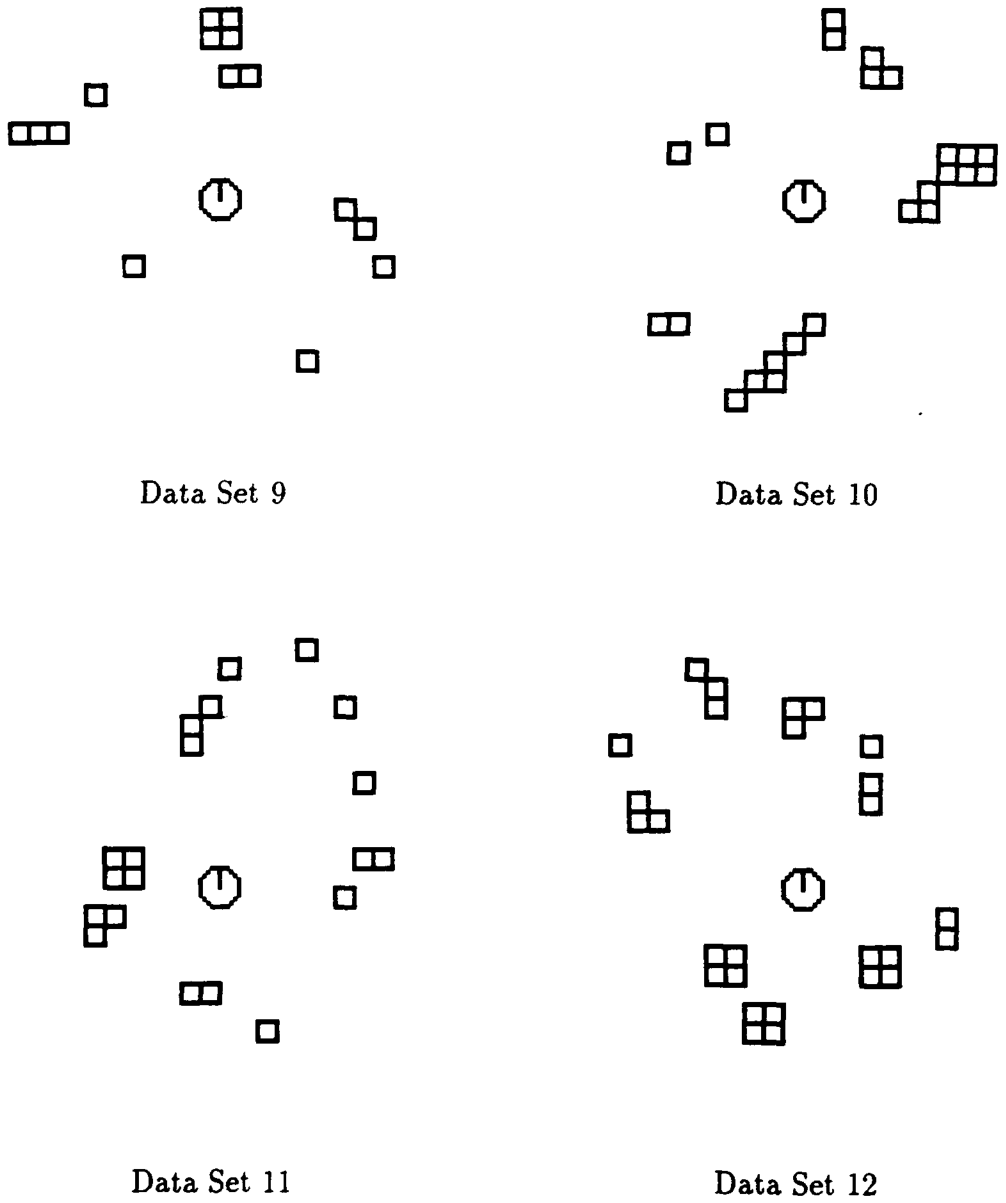
Data Set 1

Data Set 2

Data Set 3

Data Set 4

Figure E-1: Room 915 data sets 1–4.



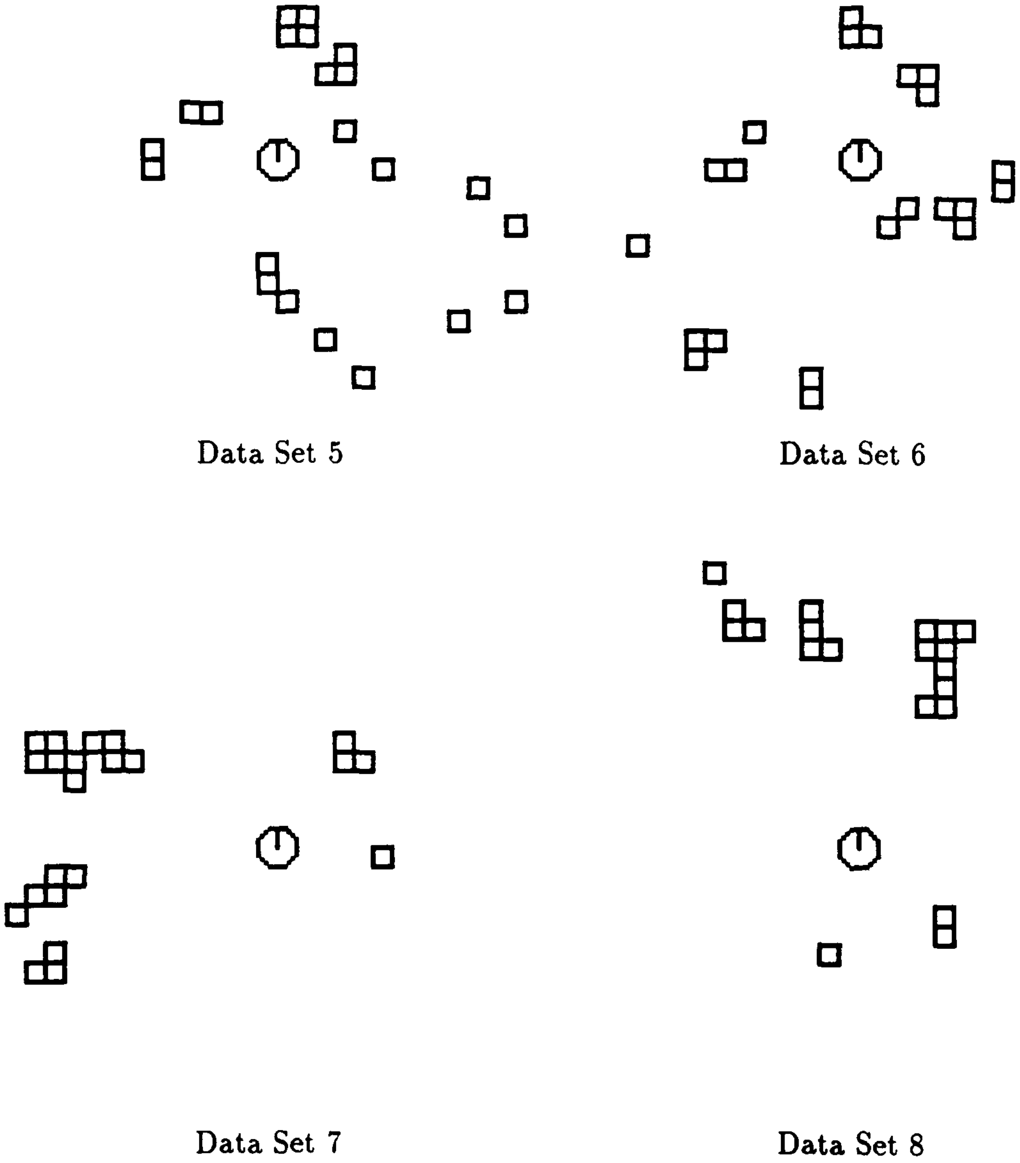
Data Set 9

Data Set 10

Data Set 11

Data Set 12

Figure E-3: Room 915 data sets 9–12.



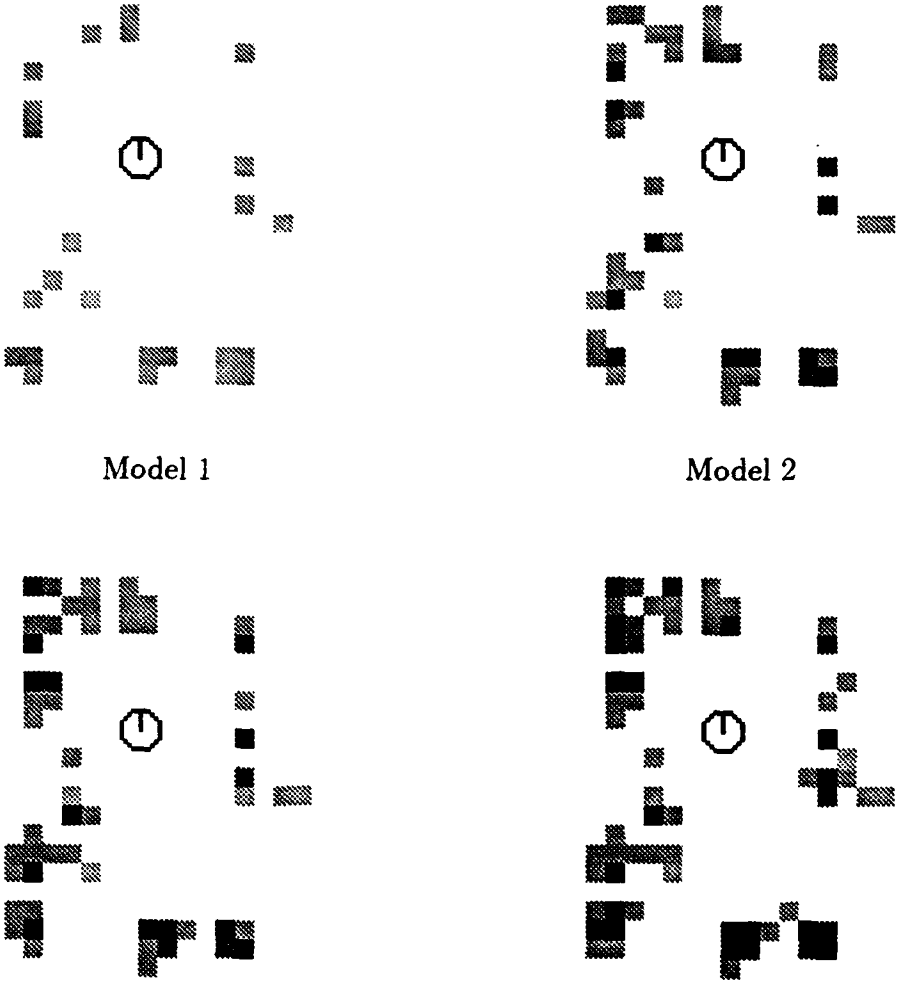
Data Set 5

Data Set 6

Data Set 7

Data Set 8

Figure E-2: Room 915 data sets 5–8.



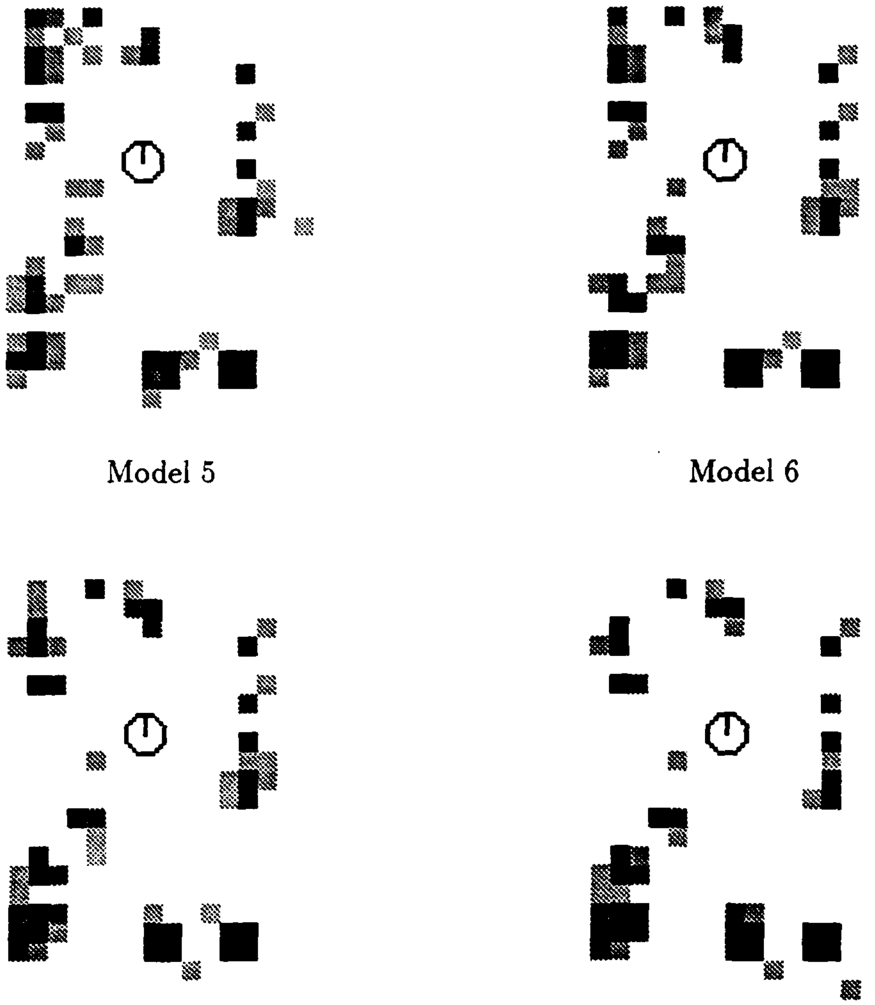
Model 1

Model 2

Model 3

Model 4

Figure E-4: Room 915 models 1–4.



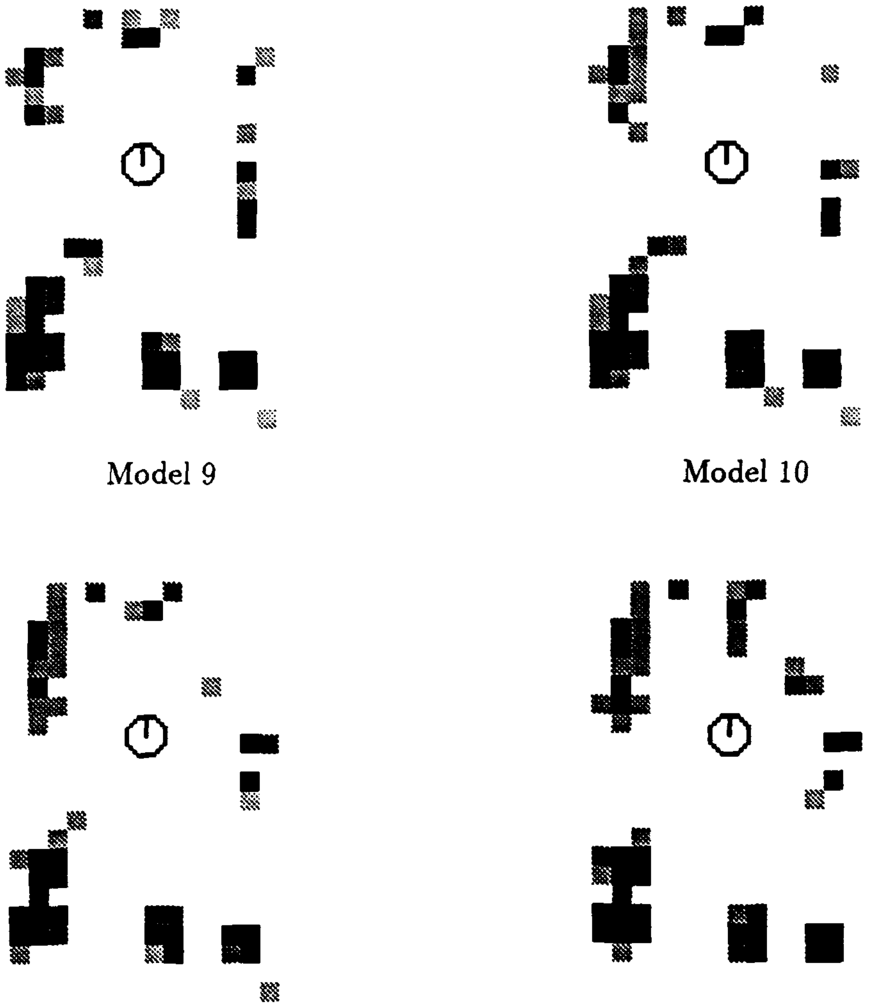
Model 5

Model 6

Model 7

Model 8

Figure E-5: Room 915 models 5–8.



Model 9

Model 10

Model 11

Model 12

Figure E-6: Room 915 models 9–12.

Appendix F

Data and Models for Room 9PR

This appendix contains a data set taken from Room 9PR (the 9^{th} floor playroom) in the MIT Artificial Intelligence Laboratory and the resulting model.

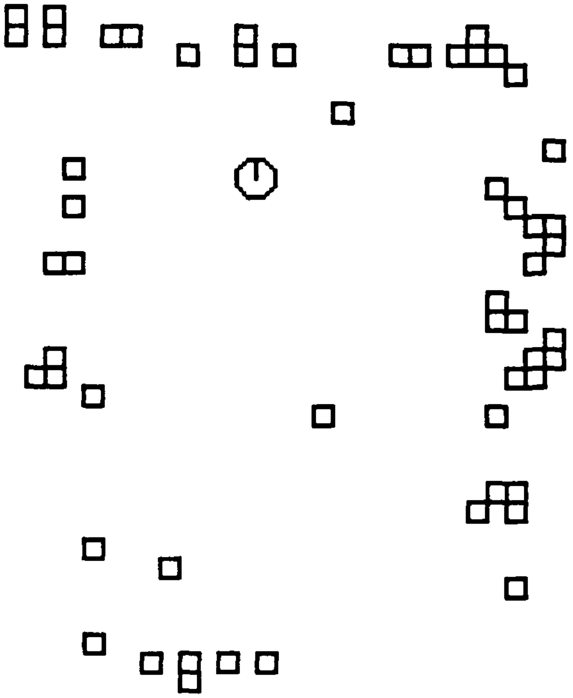


Figure F-1: Room 9PR data set.

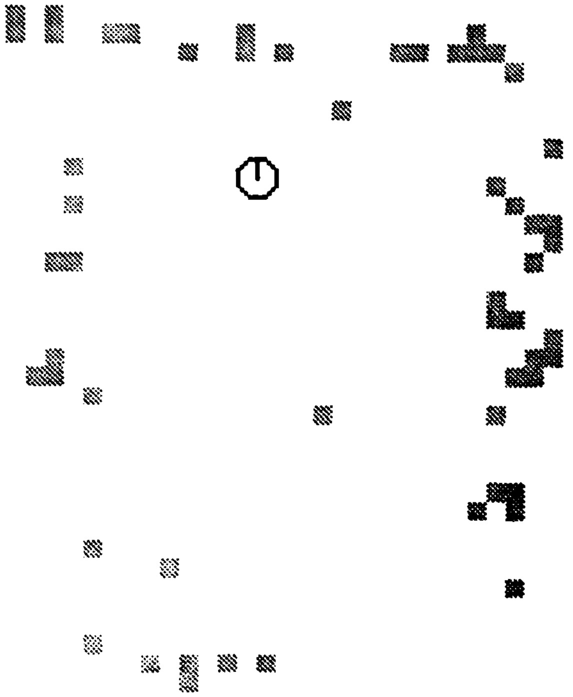
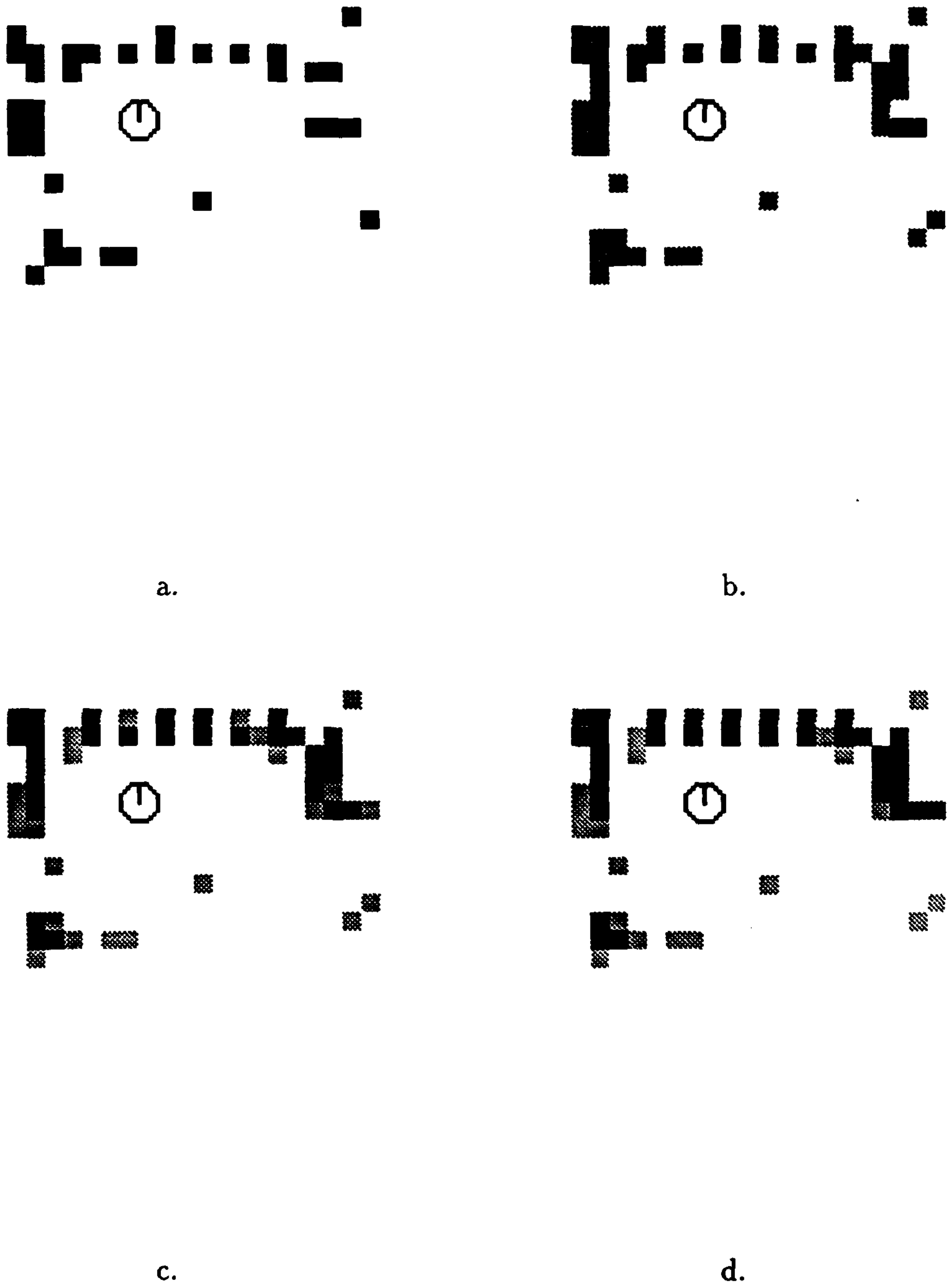


Figure F-2: Room 9PR model.

Appendix G

Room 914 Linear Model Update Examples

This appendix shows the effect of varying the parameters of the linear model update algorithm described in Chapter 7. The figures show the final Room 914 model (Model 12) built from the data sets of Appendix D. For the models shown in the figures, the rise time t_r was varied between 1 and 8. Each figure shows the models for a single hysteresis h, ranging from 1 to 5.



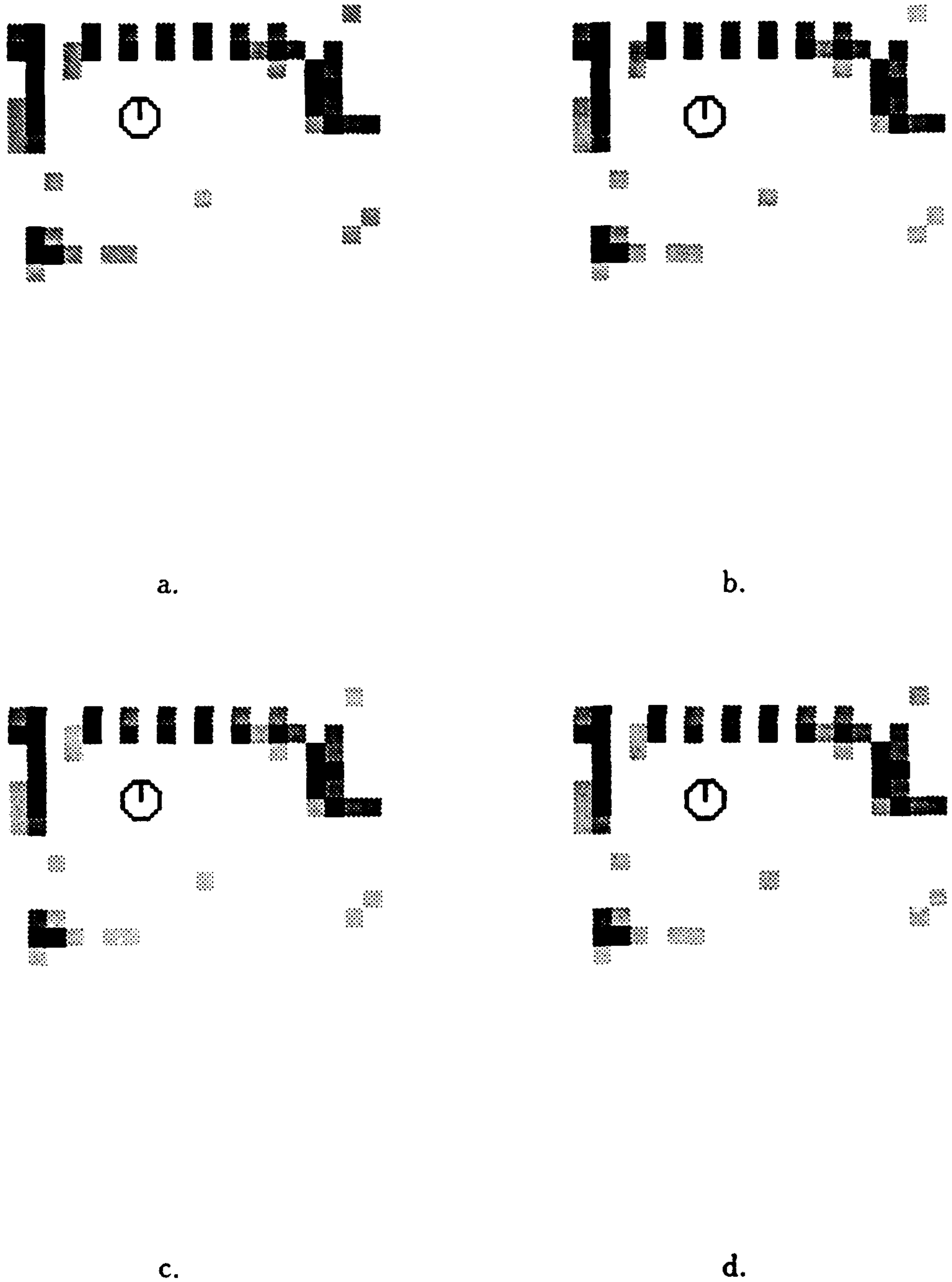
a.

b.

c.

d.

Figure G-1: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 1 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4



a.

b.

c.

d.

Figure G-2: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 1 and various rise times.
a. Rise Time = 5   b. Rise Time = 6   c. Rise Time = 7   d. Rise Time = 8

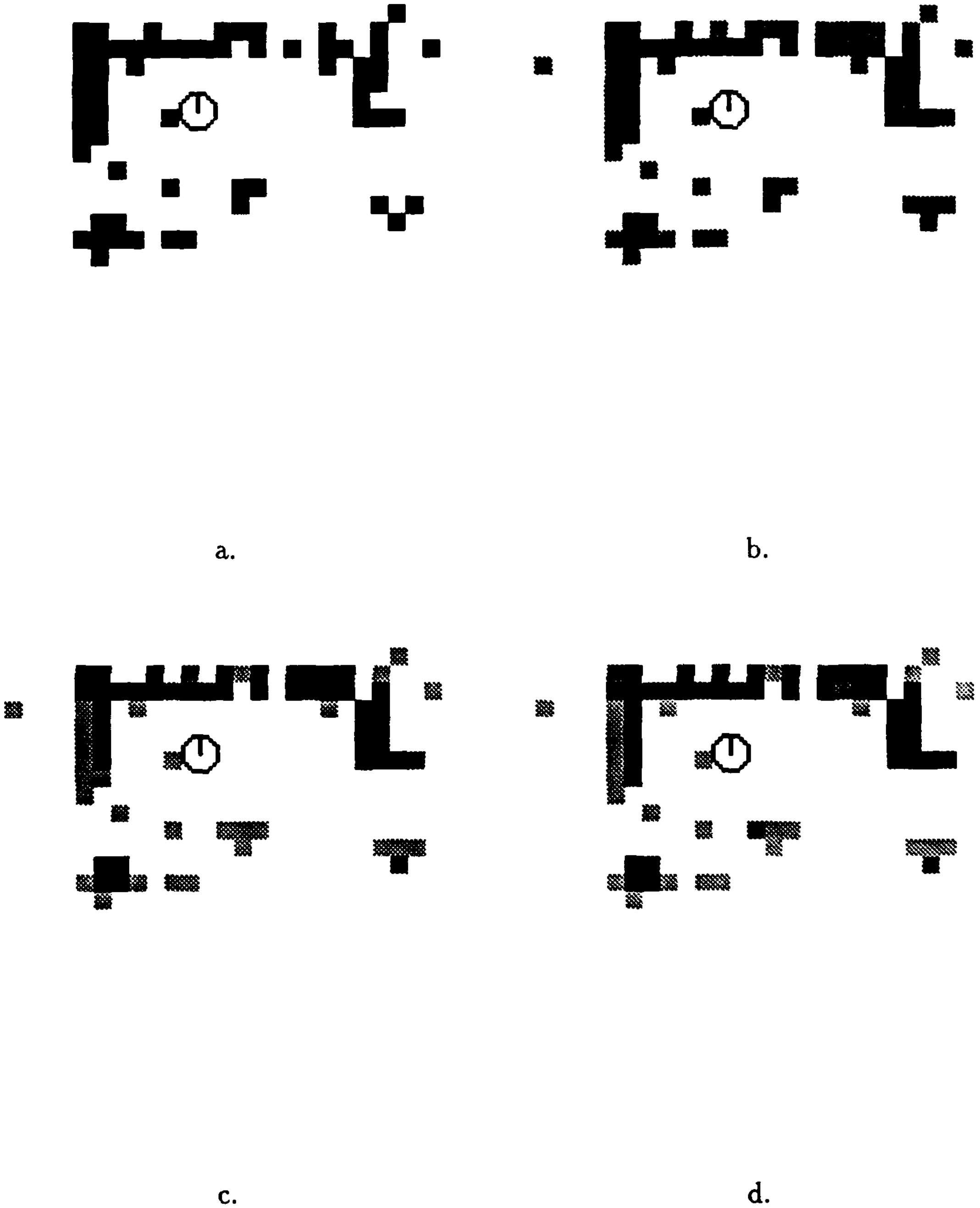
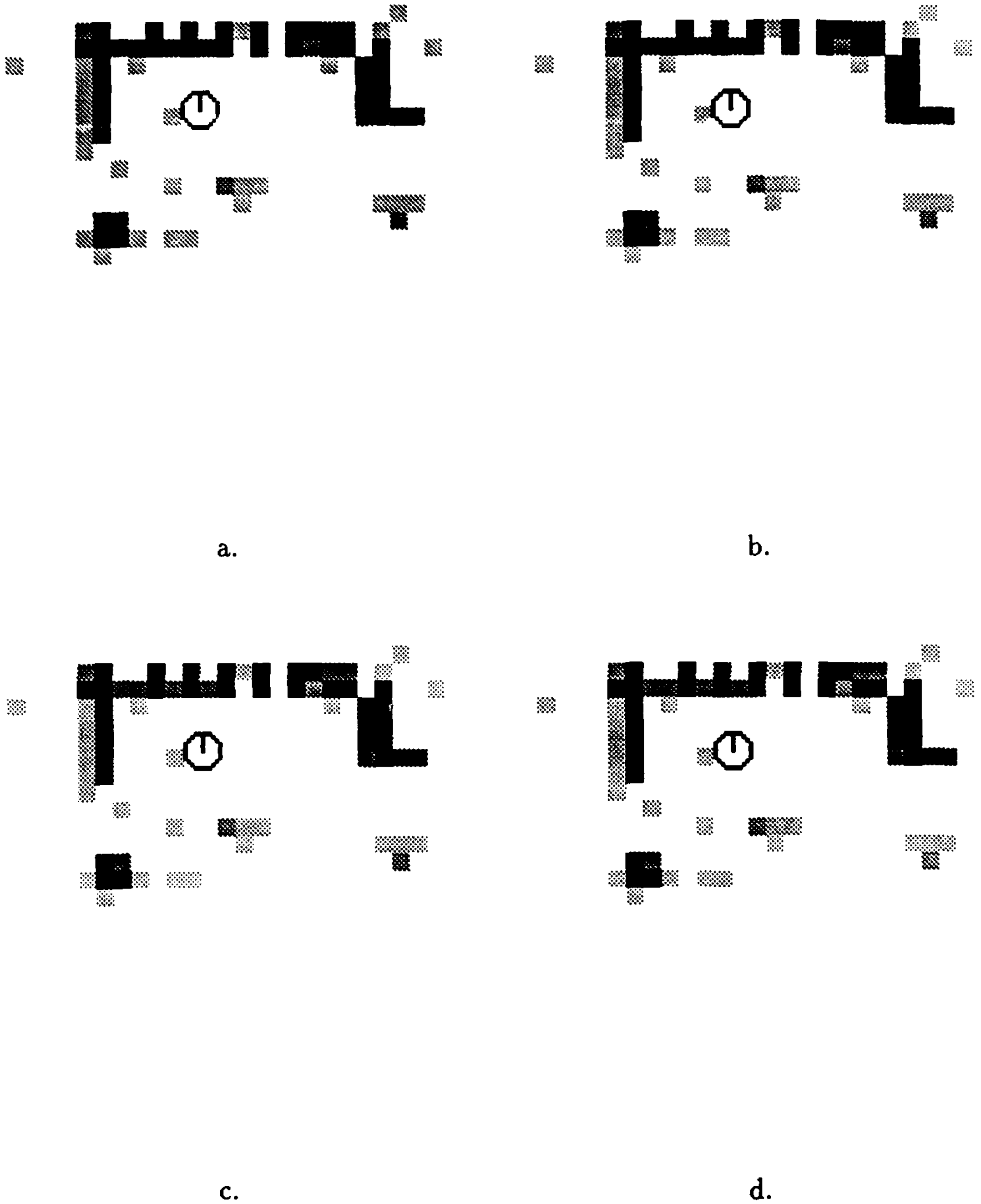


Figure G-3: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 2 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4



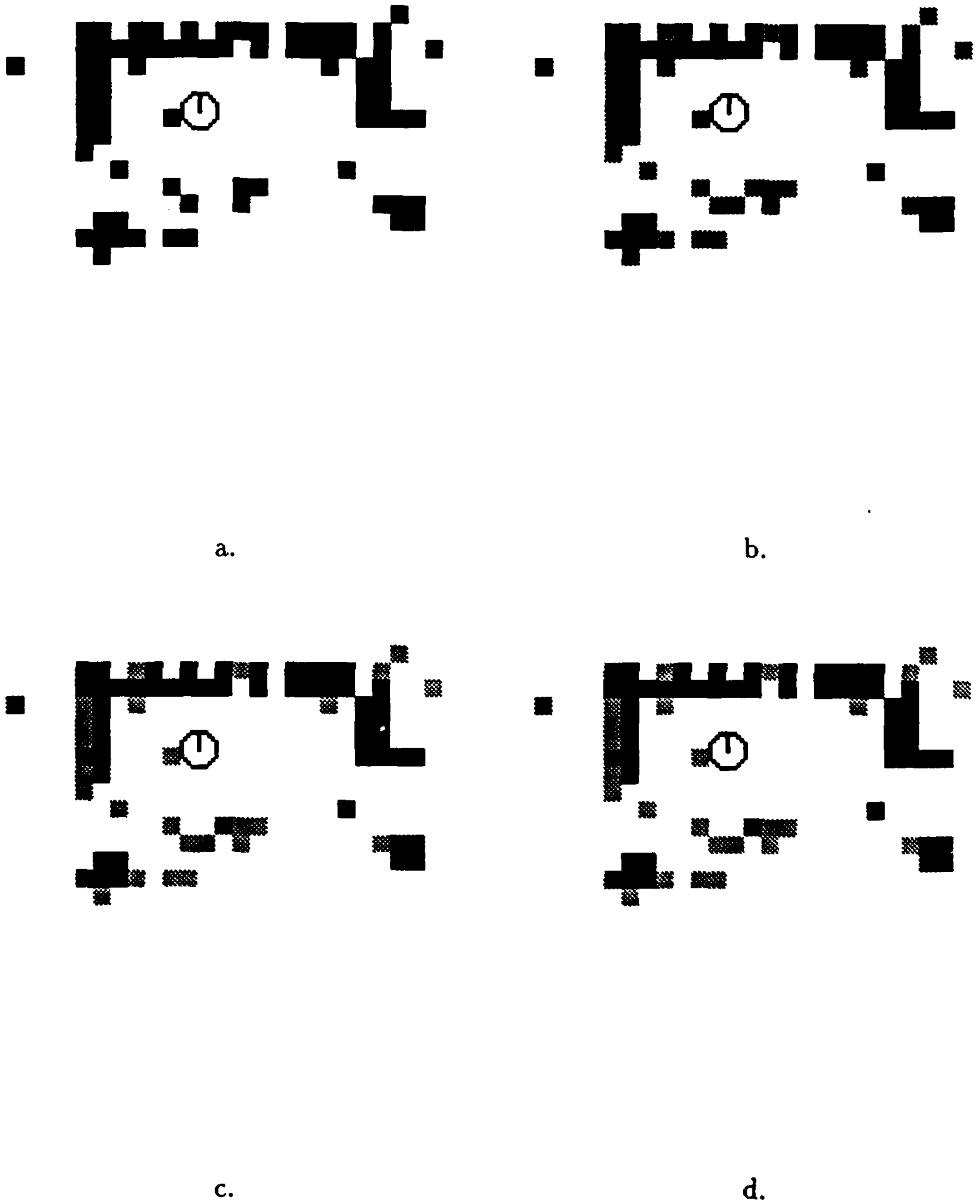
a.

b.

c.

d.

Figure G-4: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 2 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8



a.

b.

c.

d.

Figure G-5: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 3 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4

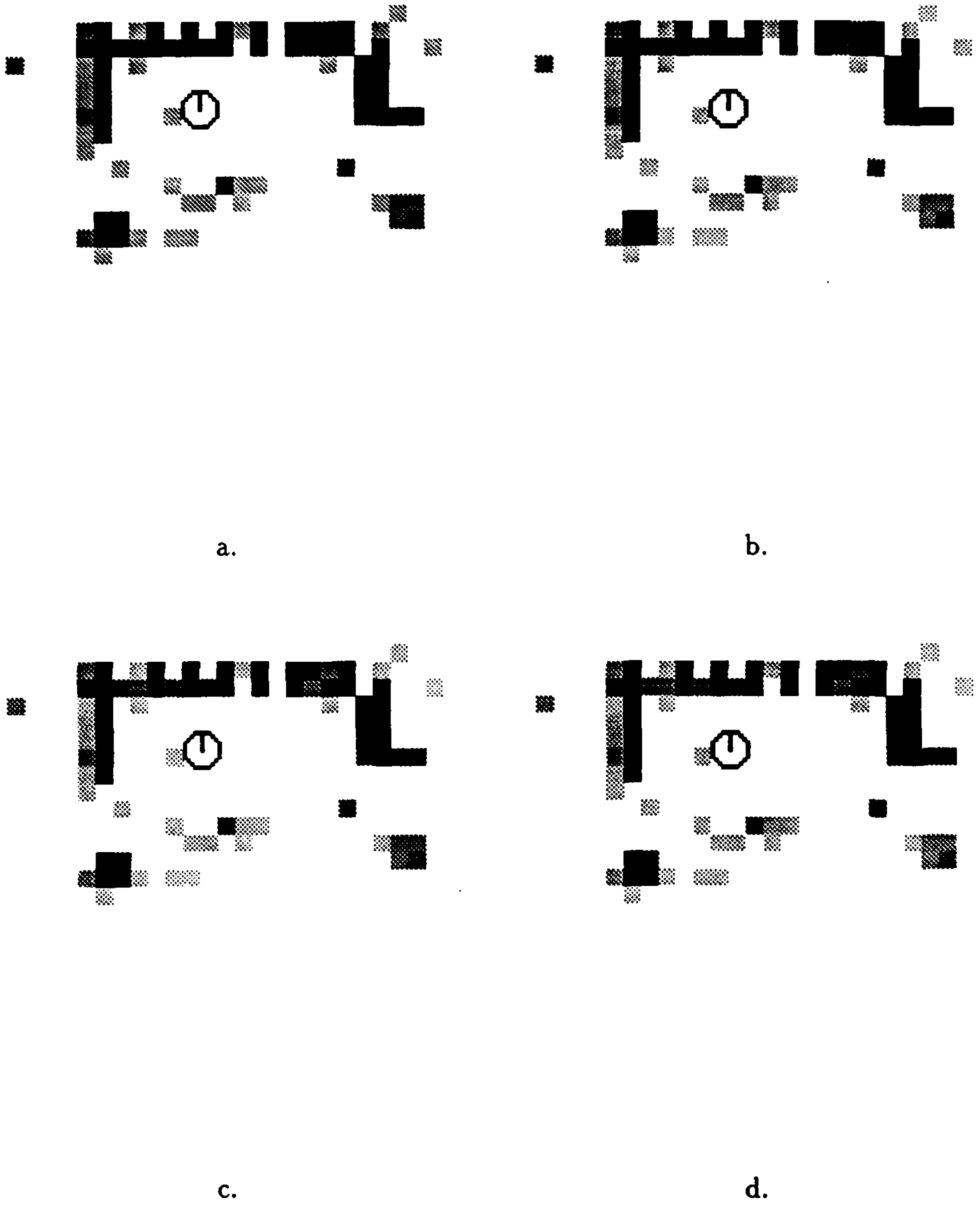


Figure G-6: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 3 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8

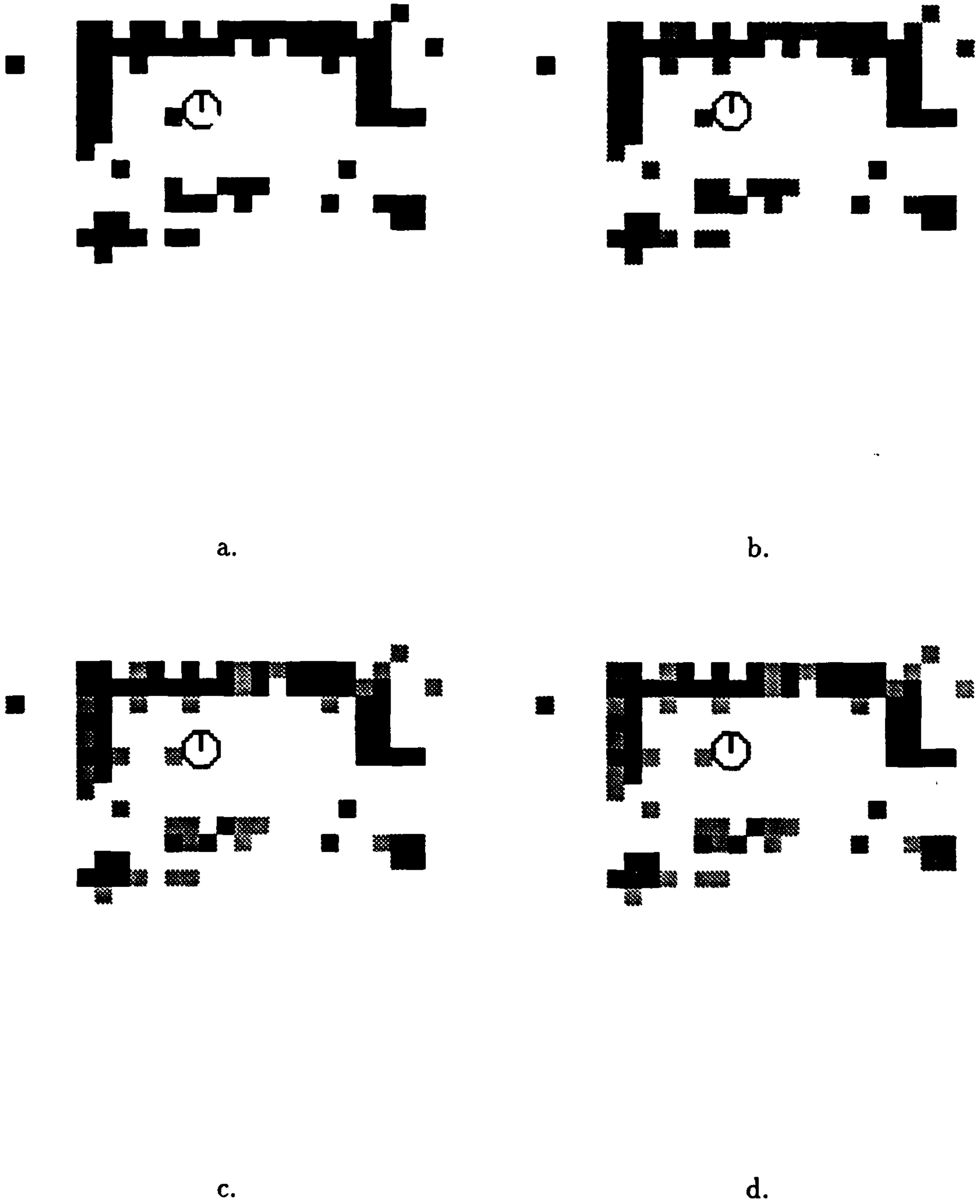


Figure G-7: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 4 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4

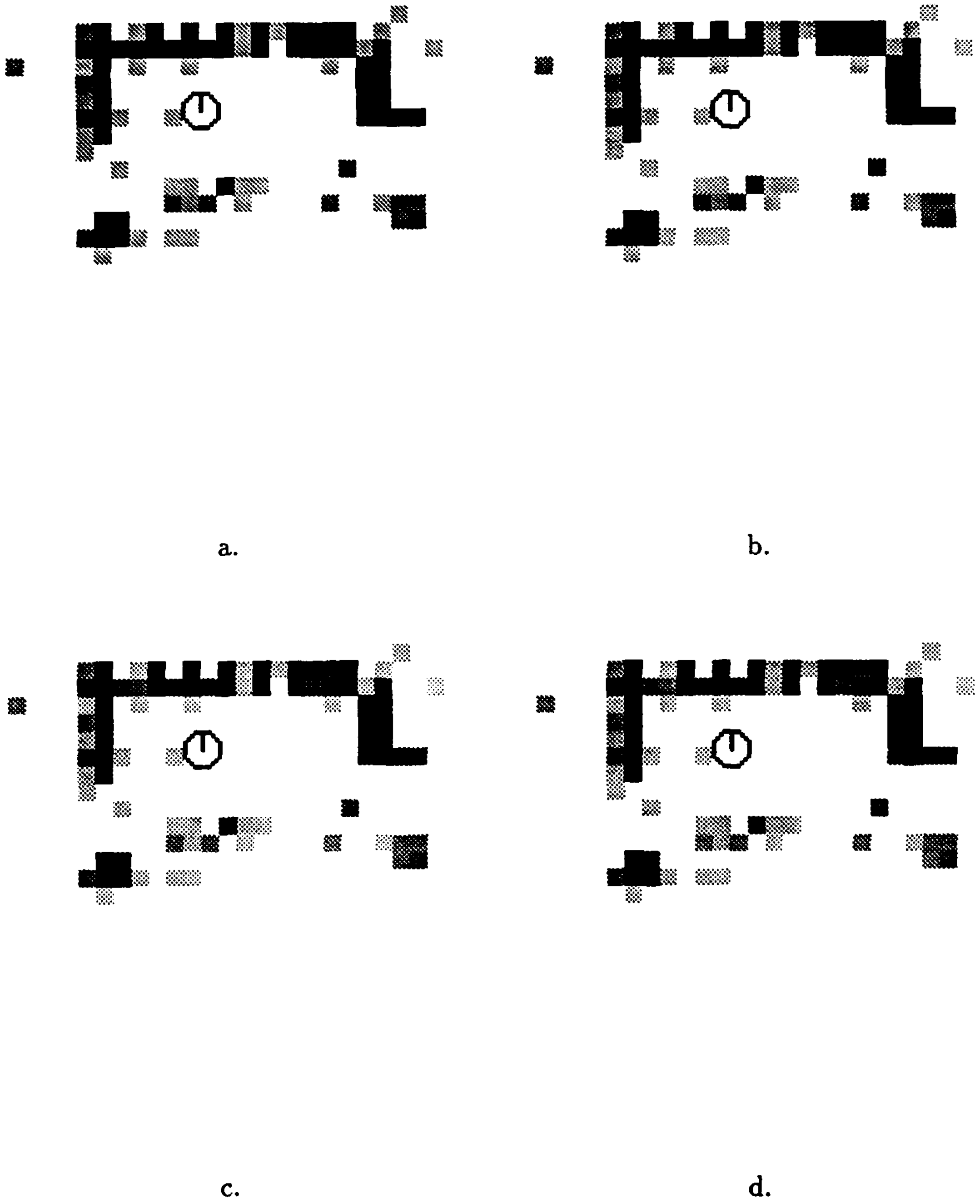
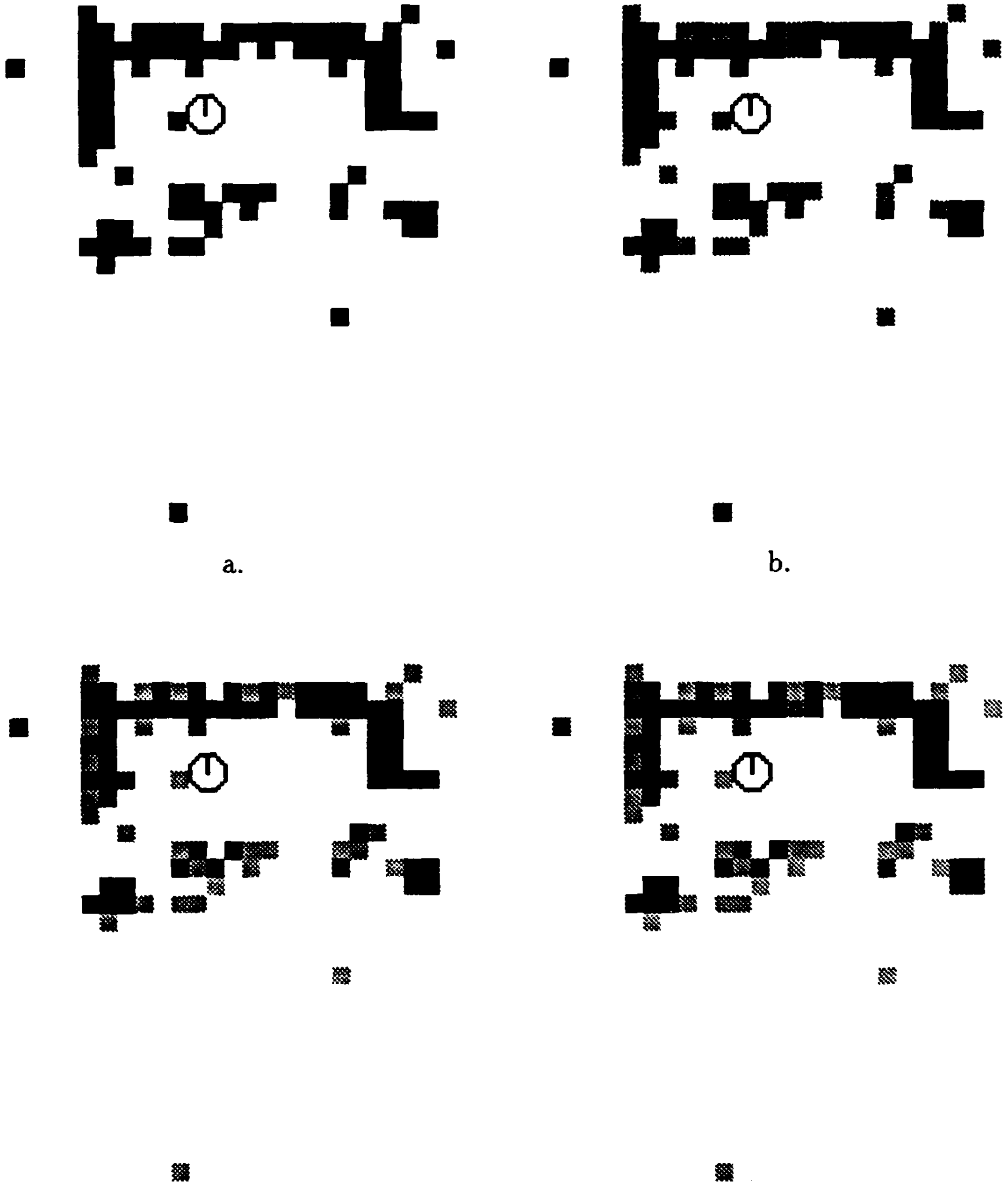
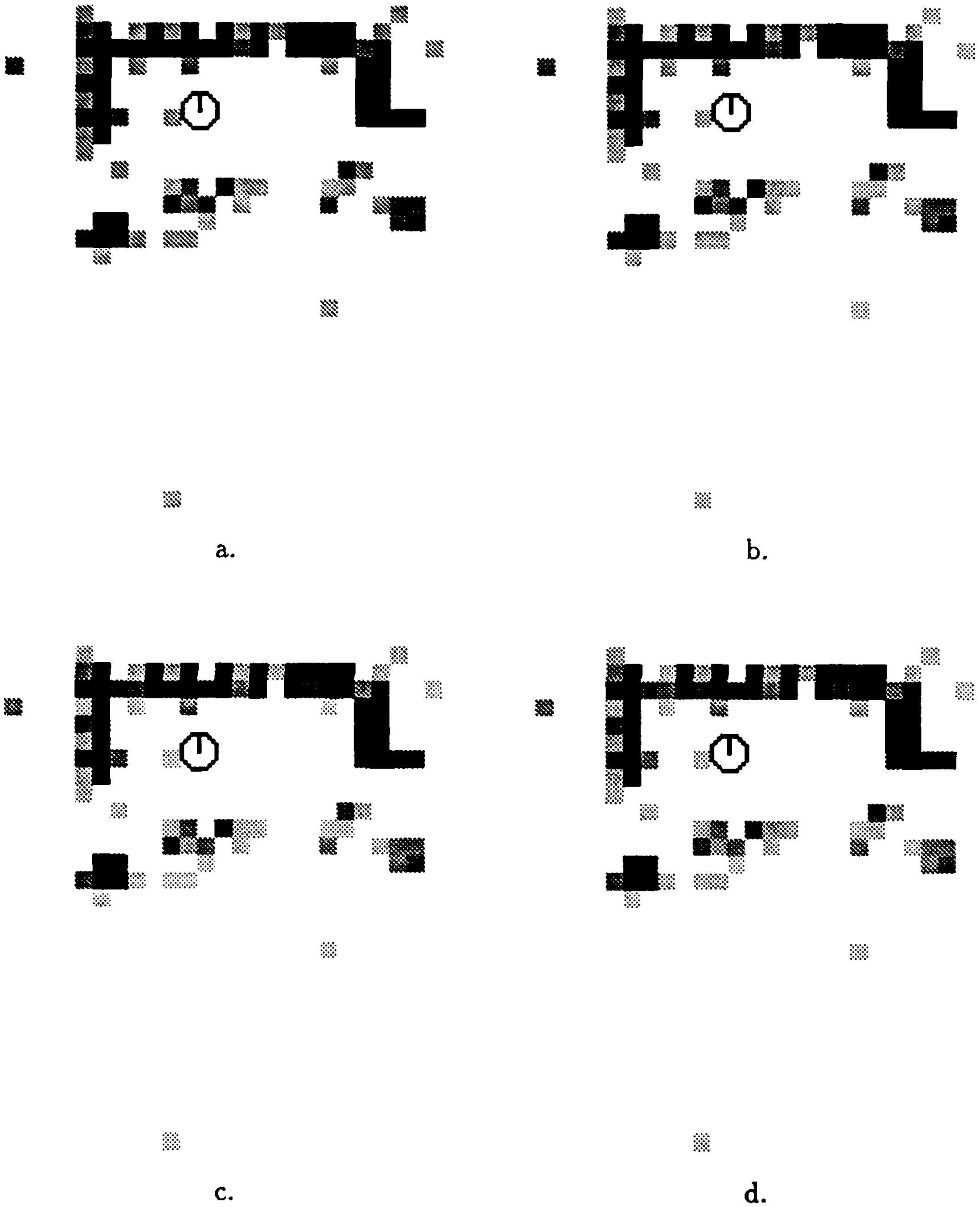


Figure G-8: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 4 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8



a. b. c. d.

Figure G-9: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 5 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4



a. b. c. d.

Figure G-10: Examples of the final Room 914 model using the linear update algorithm with Hysteresis = 5 and various rise times.
a. Rise Time = 5   b. Rise Time = 6   c. Rise Time = 7   d. Rise Time = 8

Appendix H

Room 914 Exponential Model Update Examples

This appendix shows the effect of varying the parameters of the exponential model update algorithm described in Chapter 7. The figures show the final Room 914 model (Model 12) built from the data sets of Appendix D. For the models shown in the figures, the rise time t_r was varied between 1 and 8. Each figure shows the models for a single hysteresis h, ranging from 1 to 5.

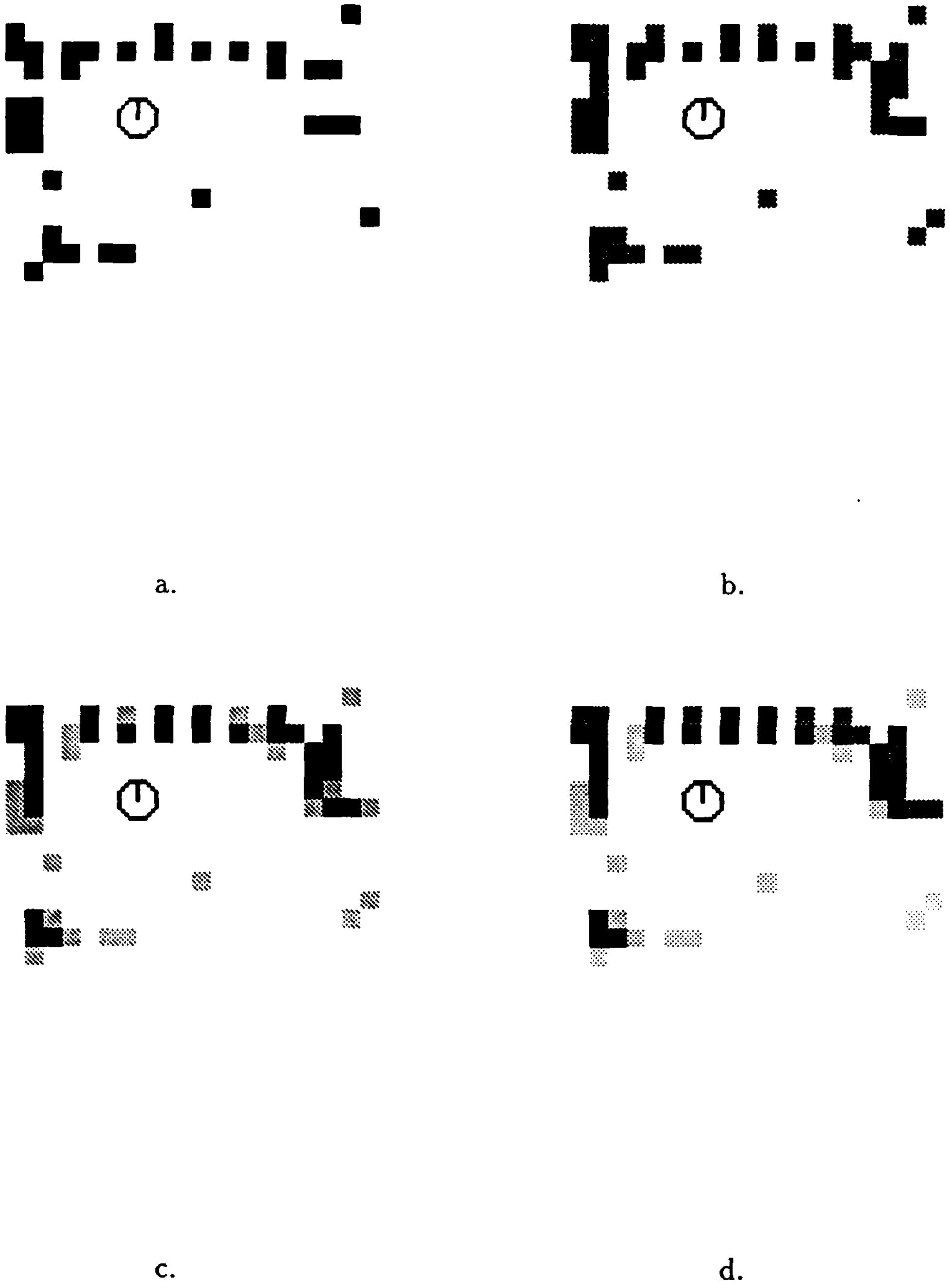


Figure H-1: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 1 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4

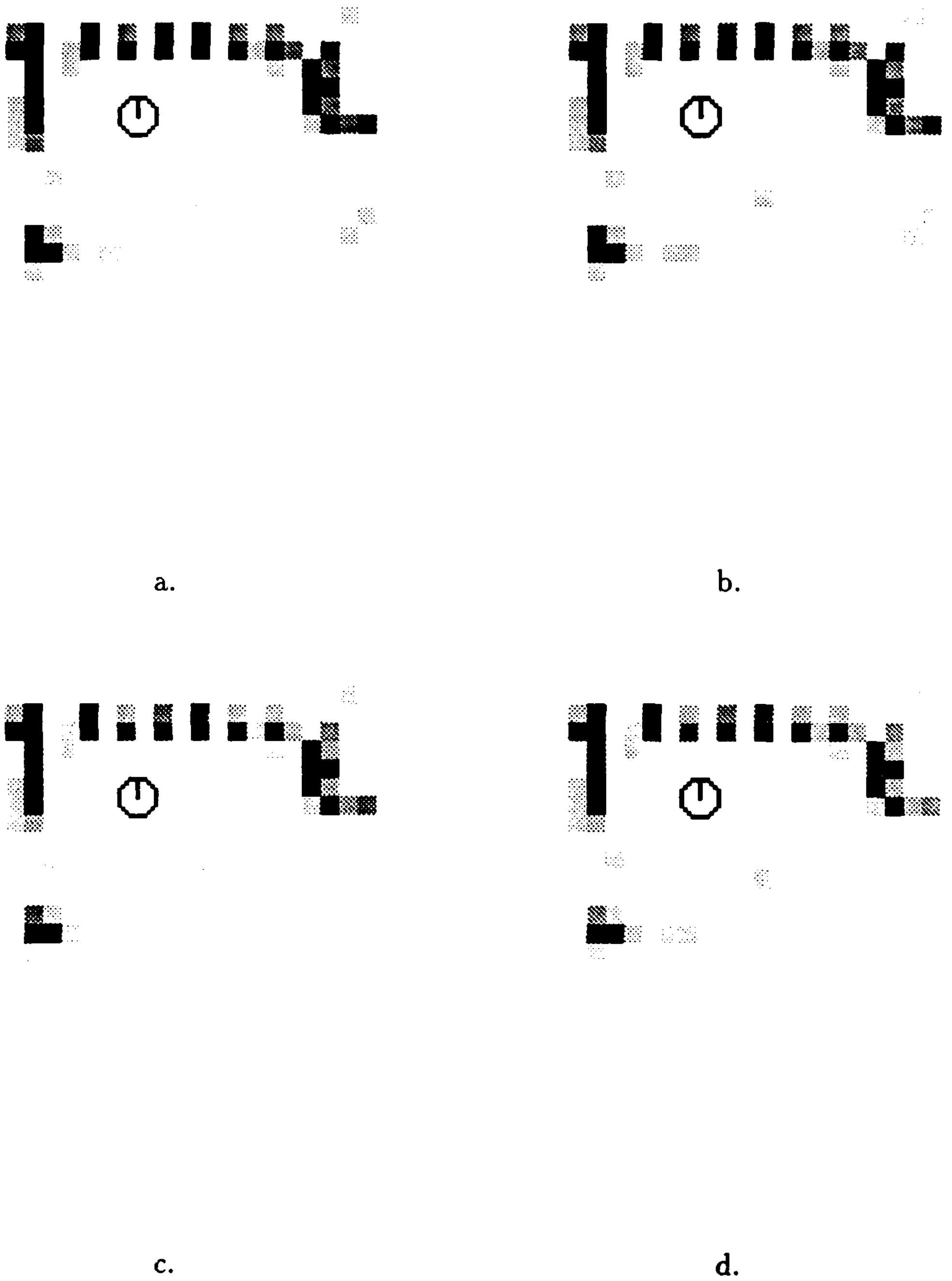


Figure H-2: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 1 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8

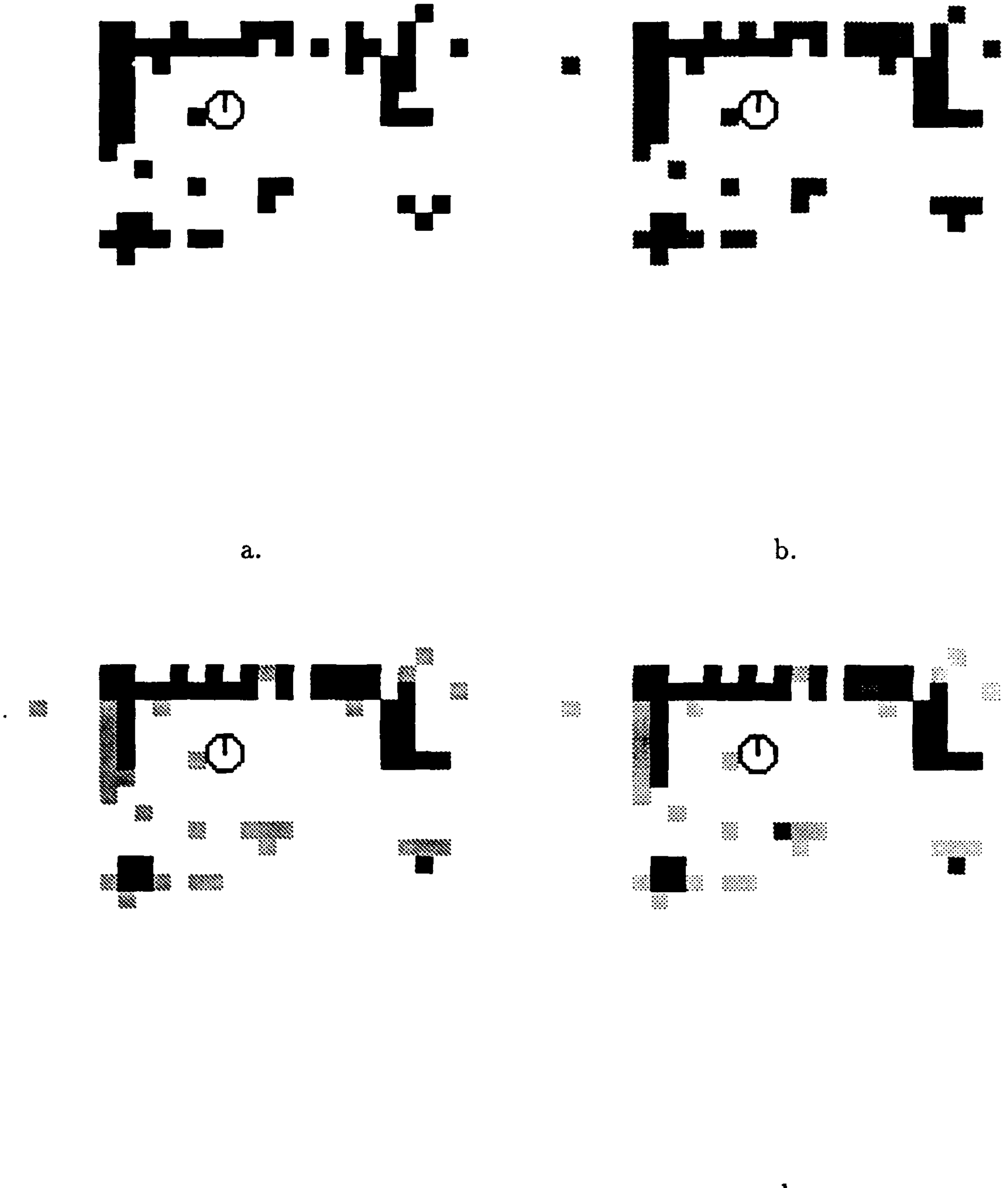
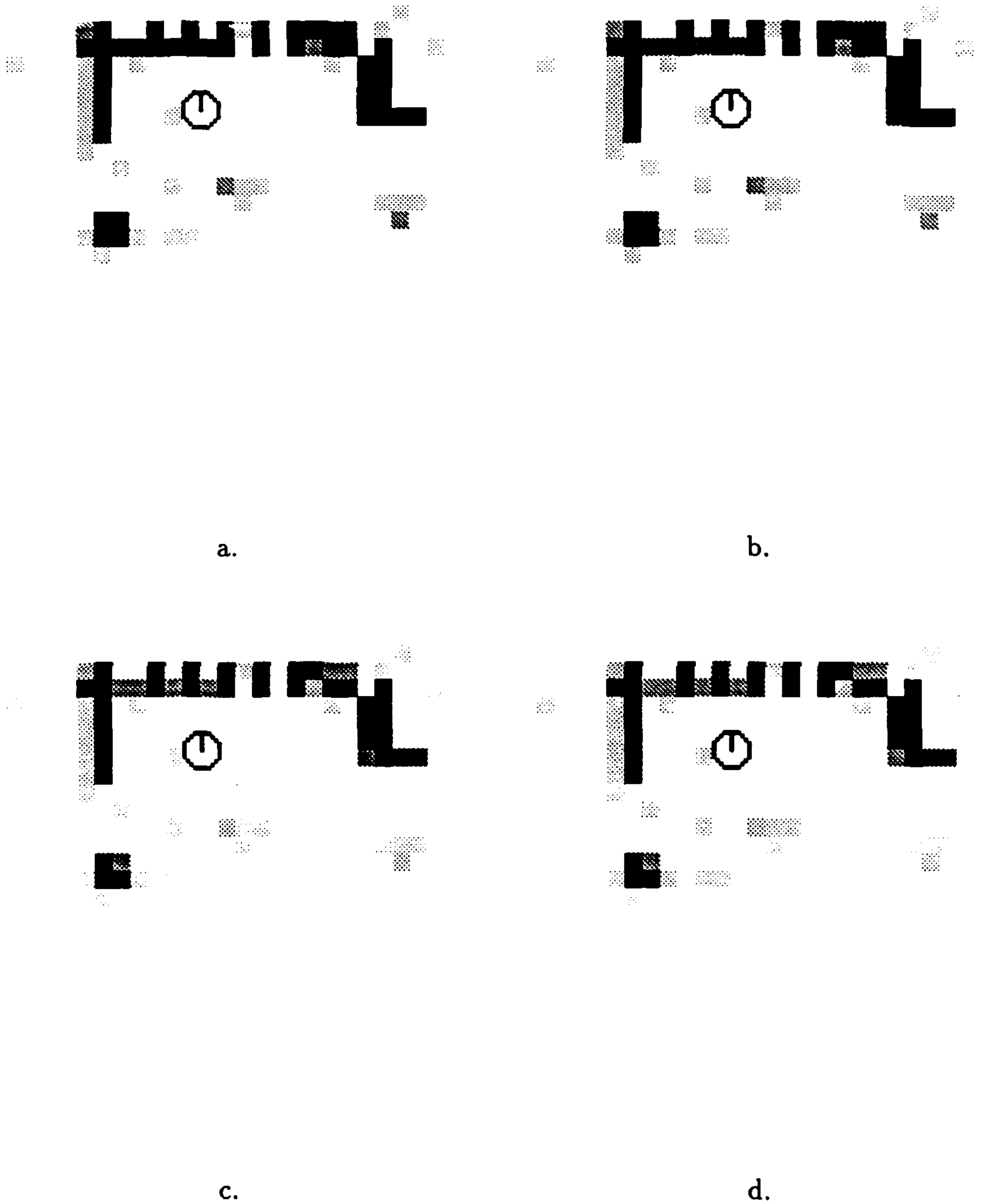


Figure H-3: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 2 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4



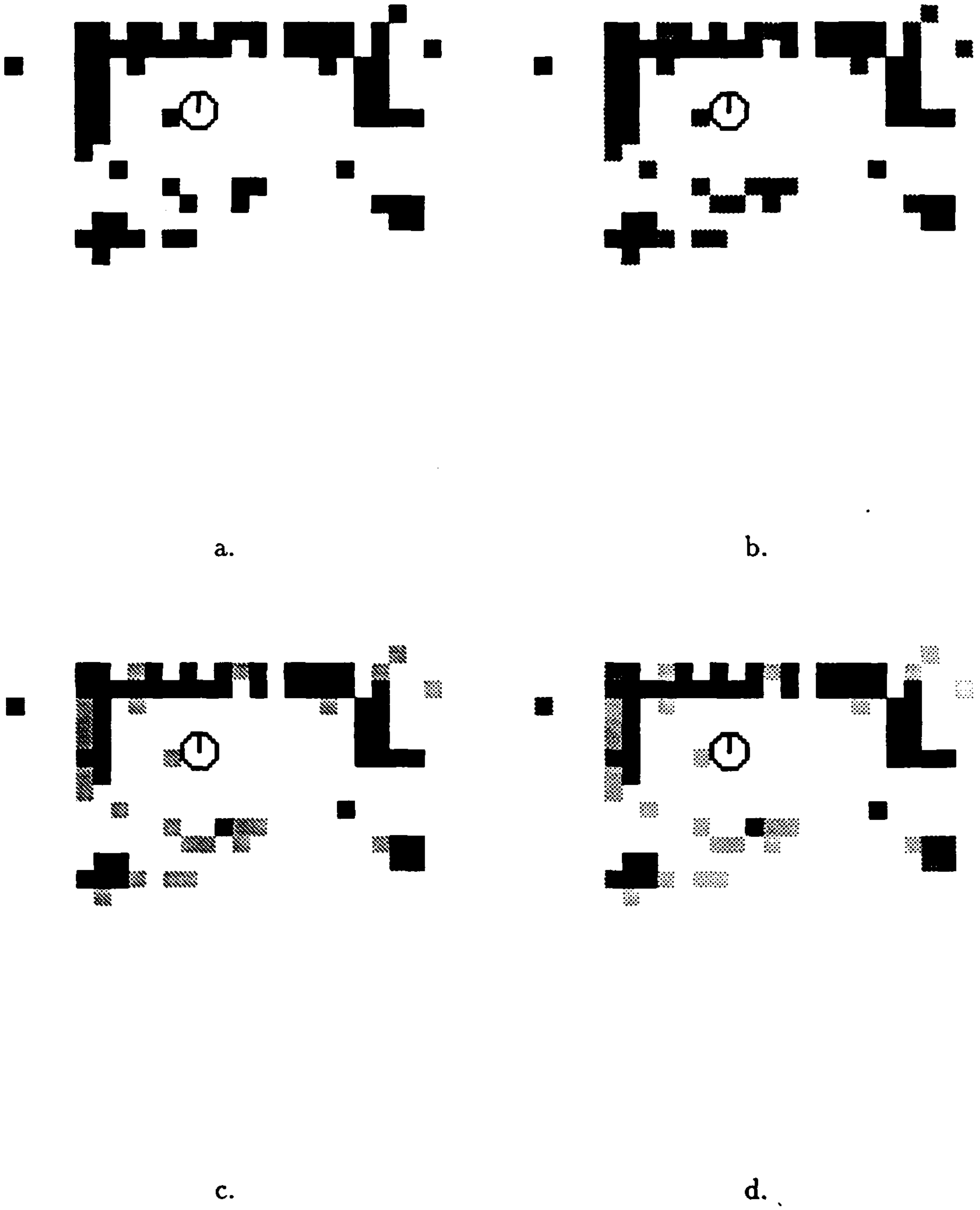
a.

b.

c.

d.

Figure H-4: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 2 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8



a.

b.

c.

d.

Figure H-5: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 3 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4

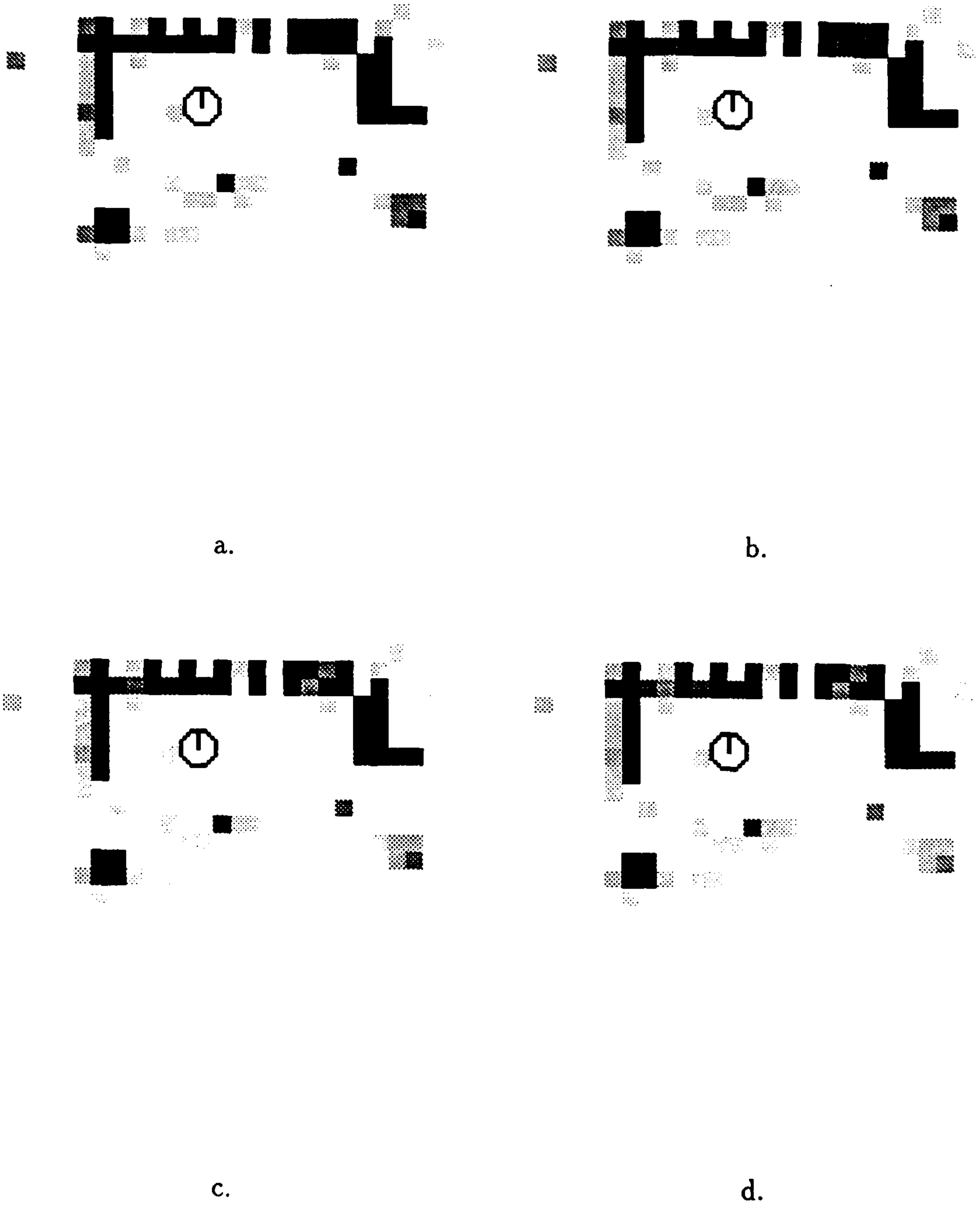


Figure H-6: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 3 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8

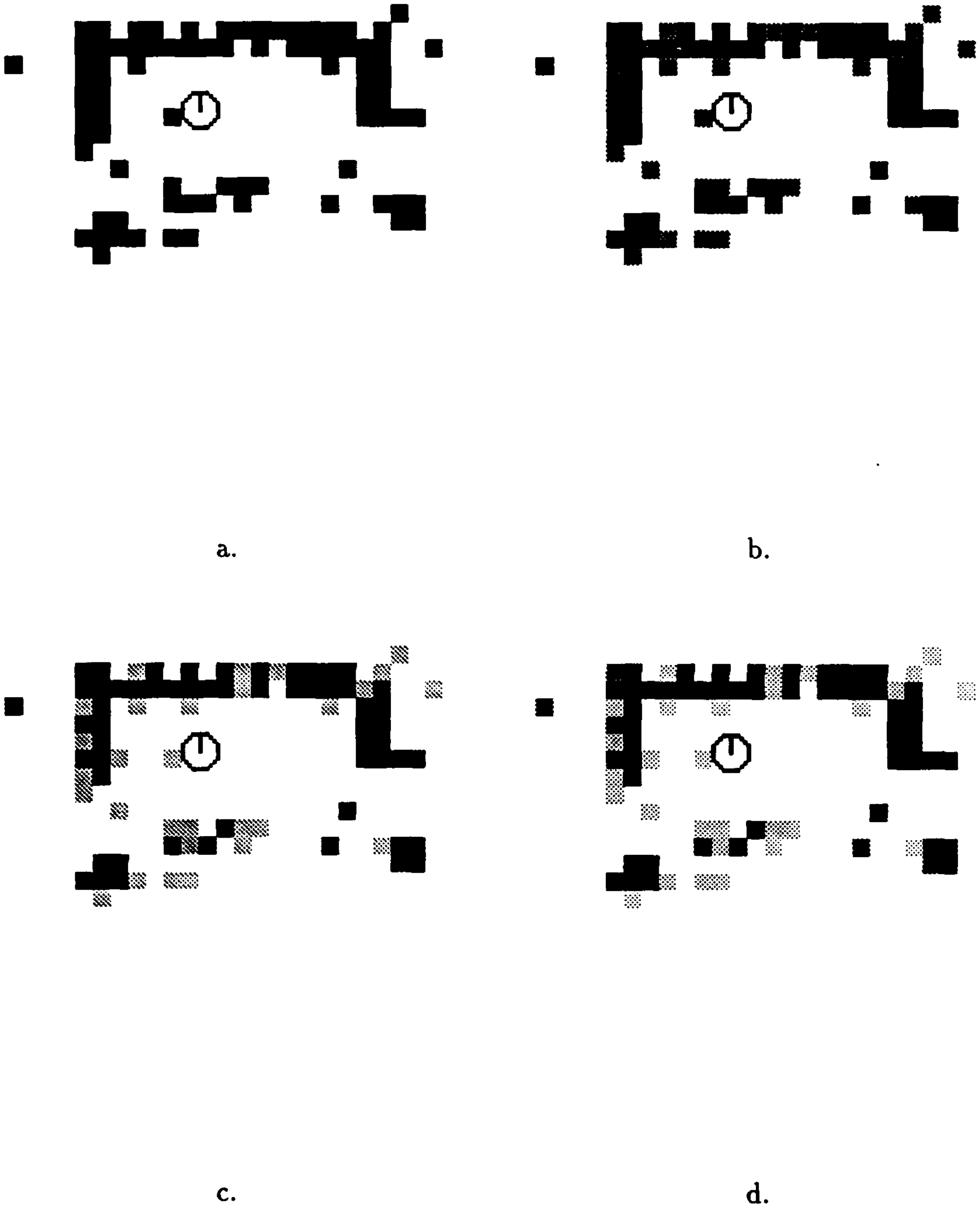


Figure H-7: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 4 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4

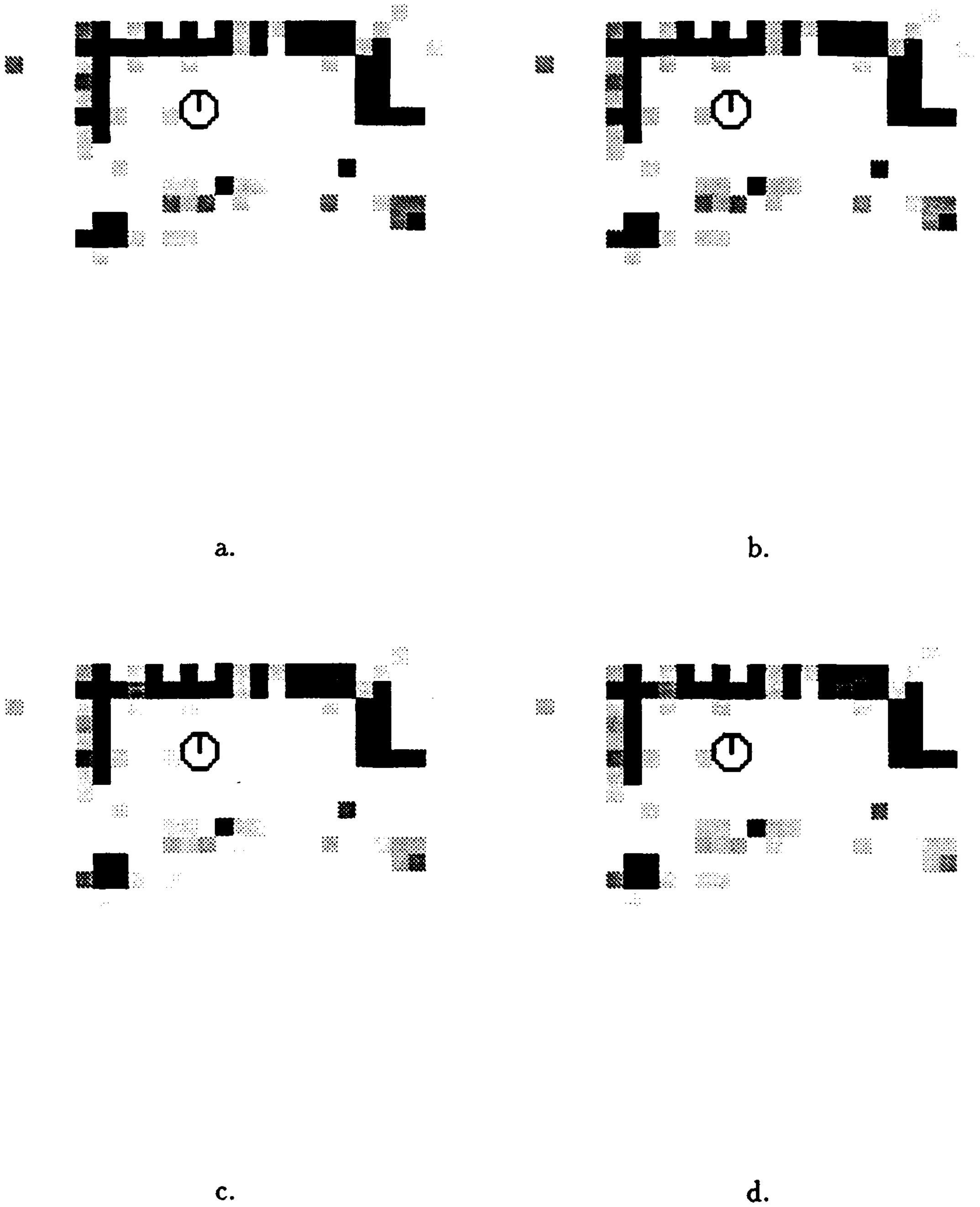


Figure H-8: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 4 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8

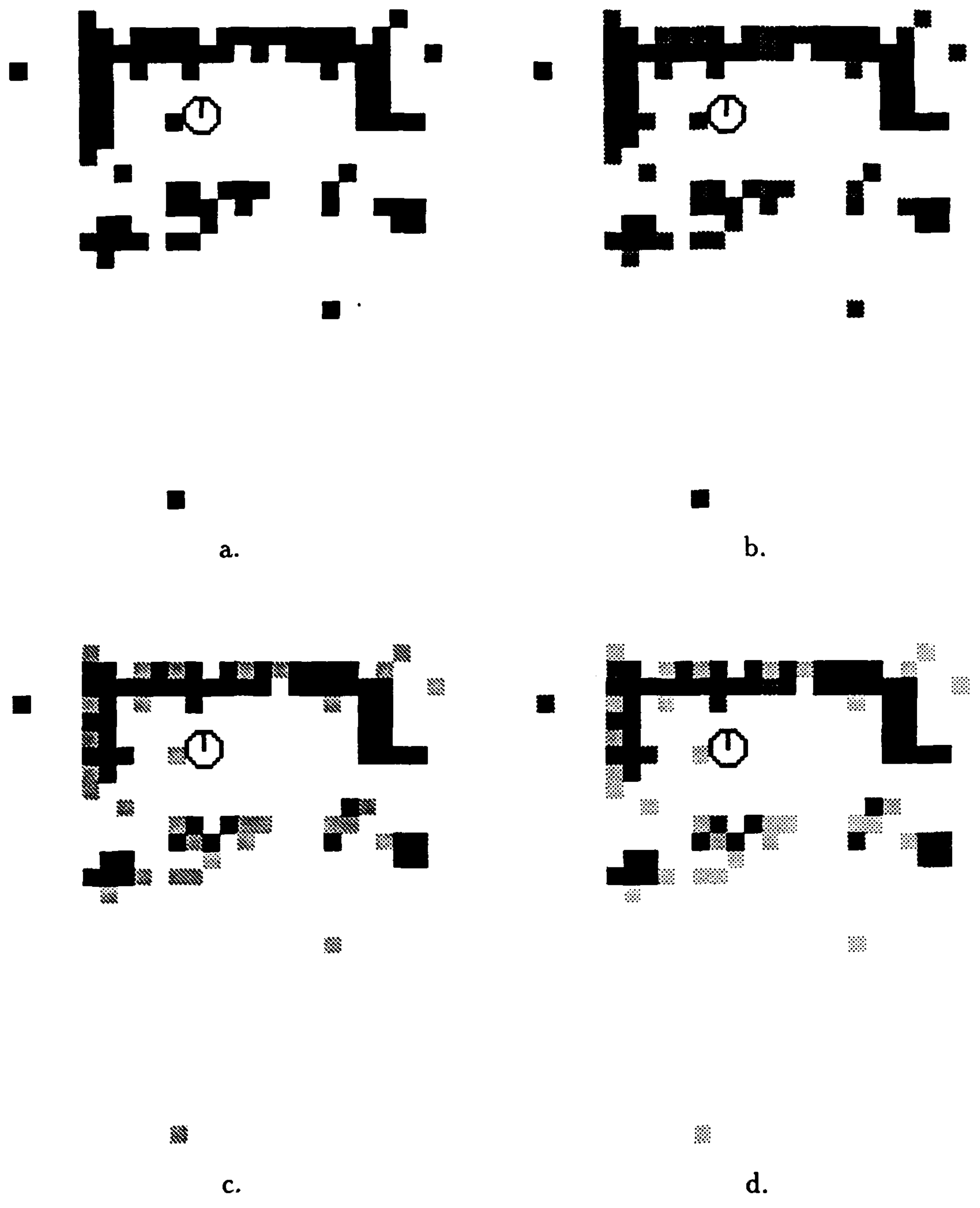
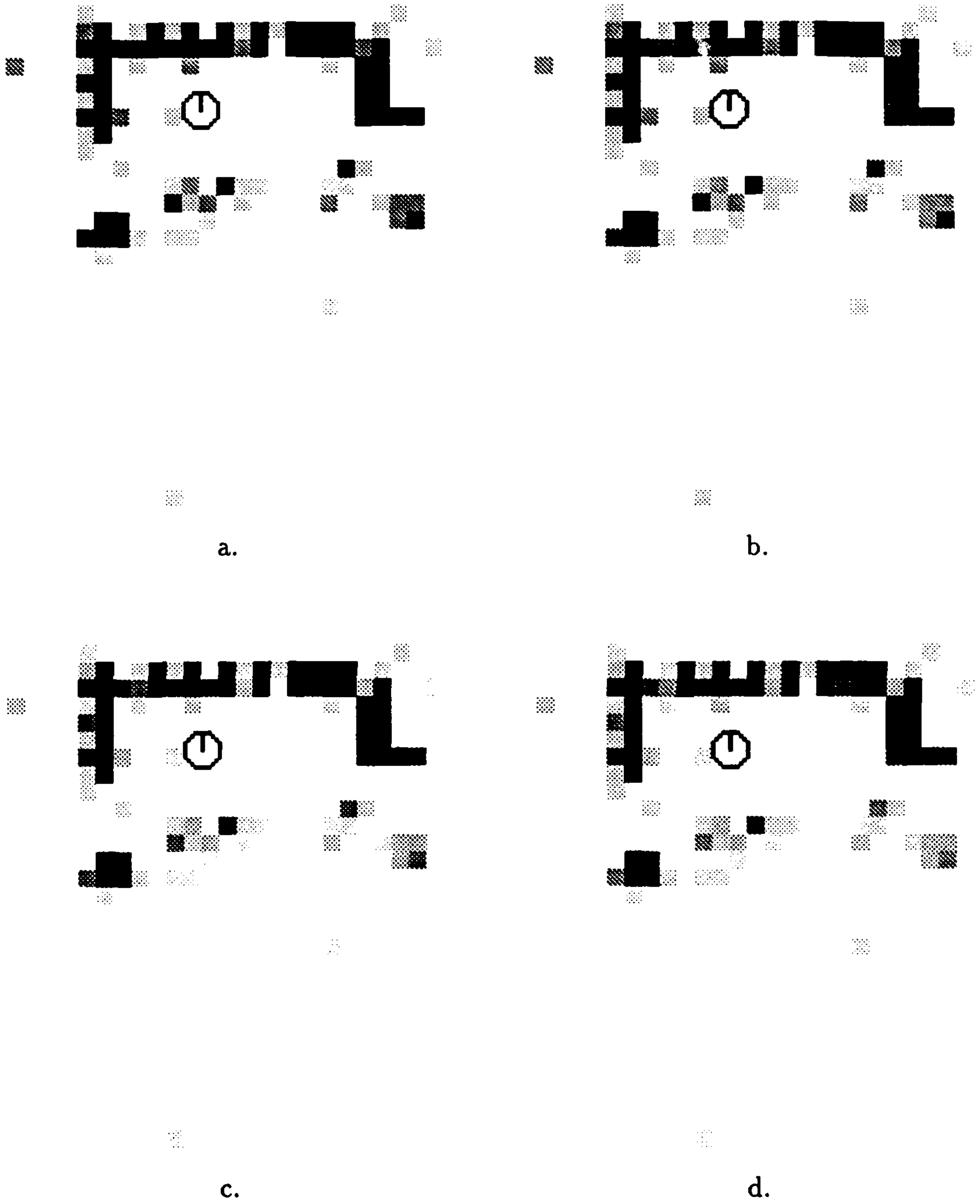


Figure H-9: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 5 and various rise times.
a. Rise Time = 1 b. Rise Time = 2 c. Rise Time = 3 d. Rise Time = 4



a.

b.

c.

d.

Figure H-10: Examples of the final Room 914 model using the exponential update algorithm with Hysteresis = 5 and various rise times.
a. Rise Time = 5 b. Rise Time = 6 c. Rise Time = 7 d. Rise Time = 8

Bibliography

Arkin, R. C. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proc. IEEE International Conference on Robotics and Automation*, 264–271, Raleigh, NC, March 1987.

Ayache, N., and O. D. Faugeras. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–8(1):44–54, January 1986.

Ayache, N., O. D. Faugeras, B. Faverjon, and G. Toscani. Matching depth maps obtained by passive stereo. In *Proc. Third Workshop on Computer Vision: Representation and Control*, 197–204, Bellaire, MI, October 1985.

Ballard, D. H., and C. M. Brown. *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall, Inc. 1982.

Baxandall, L. *World Guide to Nude Beaches and Recreation.* New York: Harmony Books. 1983.

Binford, T. O. Survey of stereo mapping systems and related and supporting techniques and literature. Computer Science Department, Stanford University, April 1982.

Boissonnat, J. D., O. D. Faugeras, and E. Le Bras-Mehlman. Representing stereo data with the Delaunay triangulation. Technical Report 788, INRIA, February 1988a.

Boissonnat, J. D., O. D. Faugeras, and E. Le Bras-Mehlman. Representing stereo data with the Delaunay triangulation. In *Proc. IEEE International Conference on Robotics and Automation*, 1798–1803, Philadelphia, PA, April 1988b.

Bolles, R. C., and R. A. Cain. Recognizing and locating partially visible objects: The local-feature-focus method. *The International Journal of Robotics Research*, 1(3):57–82, Fall 1982.

Braunegg, D. J. An approach to industrial computer vision using syntactic/semantic learning techniques. Master's thesis, University of Tennessee, August 1983.

Braunegg, D. J. Stereo feature matching in disparity space. AI Memo 1184, MIT, September 1989a.

Braunegg, D. J. An alternative to using the 3-D Delaunay tessellation for representing freespace. AI Memo 1185, MIT, September 1989b.

Braunegg, D. J. Location recognition using stereo vision. AI Memo 1186, MIT, October 1989c.

Braunegg, D. J. Stereo feature matching in disparity space. In *Proc. IEEE International Conference on Robotics and Automation*, Cincinnati, OH, May 1990.

Braunegg, D. J., and R. C. Gonzales. An approach to industrial computer vision using syntactic/semantic learning techniques. In *Proc. Seventh International Conference on Pattern Recognition*, 1366–1369, Montreal, Canada, July 1984.

Brooks, R. A. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):190–197, March/April 1983.

Brooks, R. A. Aspects of mobile robot visual map making. In *Proc. Second International Symposium on Robotics Research*, H. Hanafusa and H. Inoue (Eds.), 325–331, Kyoto, Japan, 1984. MIT Press.

Brooks, R. A. Visual map making for a mobile robot. In *Proc. IEEE International Conference on Robotics and Automation*, 824–829, St. Louis, MO, March 1985.

Brooks, R. A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1), April 1986.

Brooks, R. A., and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(2):190–197, March/April 1985.

Burt, P., and B. Julesz. A disparity gradient limit for binocular fusion. *Science*, 208:615–617, 1980a.

Burt, P., and B. Julesz. Modifications of the classical notion of Panum's fusional area. *Perception*, 9(6):671–682, 1980b.

Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–8(6):679–698, November 1986.

Chatila, R., and J.-P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proc. IEEE International Conference on Robotics and Automation*, 138–145, St. Louis, MO, March 1985.

Connell, J. H. A colony architecture for an artificial creature. Technical Report AI-TR-1151, MIT, 1989.

de Saint Vincent, A. R. A 3D perception system for the mobile robot Hilare. In *Proc. IEEE International Conference on Robotics and Automation*, 850–855, San Francisco, CA, April 1986.

de Saint Vincent, A. R. Visual navigation for a mobile robot: Building a map of the occupied space from sparse 3-D, stereo data. In *Proc. IEEE International Conference on Robotics and Automation*, 1429–1435, Raleigh, NC, March 1987.

Deriche, R. Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87, January 1990.

Drumheller, M. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–9(2):325–332, March 1987.

Drumheller, M., and T. Poggio. On parallel stereo. In *Proc. IEEE International Conference on Robotics and Automation*, 1439–1448, San Francisco, CA, April 1986.

Faugeras, O. D., N. Ayache, and B. Faverjon. Building visual maps by combining noisy stereo measurements. In *Proc. IEEE International Conference on Robotics and Automation*, 1433–1438, San Francisco, CA, April 1986.

Flynn, A. personal communication. MIT AI Lab, 1990.

Garey, M. R., and D. S. Johnson. *Computers and Intractability (A Guide to the Theory of NP-Completeness).* New York: W. H. Freeman. 1979.

Giralt, G., R. Chatila, and M. Vaisset. An integrated navigation and motion control system for autonomous mulitsensory mobile robots. In *Proc. First International Symposium on Robotics Research*, M. Brady and R. Paul (Eds.), 191–214, Bretton Woods, NH, 1983. MIT Press.

Grimson, W. E. L. *From Images to Surfaces: A Computational Study of the Human Early Visual System.* Cambridge, MA: MIT Press. 1981.

Grimson, W. E. L. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–7(1):17–34, January 1985.

Grimson, W. E. L. personal communication, 1989.

Horn, B. K. P. *Robot Vision.* Cambridge, MA: MIT Press. 1986.

Hough, P. V. C. Method and means for recognizing complex patterns. U.S. Patent 3,069,654, 1962.

Jacobs, D. W. The use of grouping in visual object recognition. Technical Report AI-TR-1023, MIT, October 1988.

Kadanoff, M. B. Prolog-based world models for mobile robot navigation. In *Proc. of the Society of Photo-Optical Instrumentation Engineers*, 1987.

Kass, M. Linear image features in stereopsis. *International Journal of Computer Vision*, 1(4):357–368, January 1988.

Kriegman, D. J., E. Triendl, and T. O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6):792–803, December 1989.

Kuan. D. T., J. C. Zamiska, and R. A. Brooks. Natural decomposition of free space for path planning. In *Proc. IEEE International Conference on Robotics and Automation*, 168–173, St. Louis, MO, March 1985.

Kuipers, B. J. Representing knowledge of large-scale space. Technical Report AI-TR-418, MIT, July 1977.

Kuipers, B. J., and Y. T. Byun. A qualitative approach to robot exploration and map-learning. In *Proc. Workshop on Spatial Reasoning and Multi-Sensor Fusion*, October 1987.

Kuipers, B. J., and Y. T. Byun. A robust, qualitative method for robot spatial learning. In *Proc. AAAI National Conference on Artificial Intelligence*, 774–779, St. Paul, MN, August 1988.

Laumond, J.-P. *Utilisation des Graphes Planaires pour l'Apprentissage de la Structure de l'Espace d'Evolution d'un Robot Mobile.* PhD thesis, l'Université Paul Sabatier de Toulouse (Sciences), March 1984.

Laumond, J.-P. A learning system for the understanding of a mobile robot environment. In *Proc. European Working Session on Learning*, Orsay, France, February 1986. Also in *Machine (and Human) Learning*, Y. Kodratoff, ed., Michael Morgan Limited, 1987.

Le Bras-Mehlman, E., M. Schmitt, O. D. Faugeras, and J. D. Boissonnat. How the Delaunay triangulation can be used for representing stereo data. In *Proc. Second International Conference on Computer Vision*, 54–63, Tampa, FL, December 1988.

Lenz, R. K., and R. Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology. In *Proc. IEEE International Conference on Robotics and Automation*, 68–75, Raleigh, NC, March 1987.

Levine, M. D., D. A. O'Handley, and G. M. Yagi. Computer determination of depth maps. *Computer Graphics and Image Processing*, 2(2):131–150, October 1973.

Lowe, D. G. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

Lozano-Pérez, T. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C(32), February 1983.

Lozano-Pérez, T., and W. E. L. Grimson. Recognition and localization of overlapping parts from sparse data. In *Proc. Second International Symposium on Robotics Research*, H. Hanafusa and H. Inoue (Eds.), 51–56, Kyoto, Japan, 1984. MIT Press.

Lozano-Pérez, T., and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

Lynch, K. *The Image of the City.* Cambridge, MA: MIT Press. 1960.

MacVicar-Whelan, P. J., and T. O. Binford. Intensity discontinuity location to subpixel precision. In *Proc. International Joint Conference on Artificial Intelligence*, 752–754, Vancouver, BC, Canada, August 1981.

Marr, D., and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B(207):187–217, 1980.

Marr, D., and T. Poggio. A theory of human stereo vision. *Proceedings of the Royal Society of London*, B(204):301–328, 1979.

Mataric, M. J. Environment learning using a distributed representation. In *Proc. IEEE International Conference on Robotics and Automation*, Cincinnati, OH, May 1990.

Matthies, L., and S. A. Shafer. Error modelling in stereo navigation. Technical Report CMU-CS-86-140, Carnegie-Mellon University, 1986.

Mayhew, J. E. W., and J. P. Frisby. The computation of binocular edges. *Perception, 9:69–86, 1980.*

Mayhew, J. E. W., and J. P. Frisby. Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, 17:349–385, 1981.

Mish, F. C. (Ed.). *Webster's Ninth New Collegiate Dictionary.* Springfield, MA: Merriam-Webster. 1986.

Moravec, H. P. The Stanford cart and the CMU rover. *Proceedings of the IEEE*, 71(7):872–884, July 1983.

Moravec, H. P., and A. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE International Conference on Robotics and Automation*, 116–121, St. Louis, MO, March 1985.

Moravec, H. P. Towards automatic visual obstacle avoidance. In *Proc. International Joint Conference on Artificial Intelligence*, Cambridge, MA, August 1977.

Nasr, H., B. Banu, and S. Schaffer. Guiding an autonomous land vehicle using knowledge-based landmark recognition. In *Proc. Image Understanding Workshop*, 432–439, Los Angeles, CA, February 1987.

Nilsson, N. J. A mobile automaton: An application of artificial intelligence techniques. In *Proc. International Joint Conference on Artificial Intelligence*, 509–520, Washington, DC, May 1969.

O'Donnell, P. A. personal communication. MIT AI Lab, 1990.

Ohta, Y., and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154, March 1985.

Pavlidis, T. *Algorithms for Graphics and Image Processing.* Rockville, MD: Computer Science Press. 1982.

Piaget, J., and B. Inhelder. *The Child's Conception of Space.* New York: Norton. 1967. First published in French, 1948.

Poggio, G. F., and T. Poggio. The analysis of stereopsis. *Annual Review of Neuroscience*, 7:379–412, 1984.

Pollard, S. B., J. E. W. Mayhew, and J. P. Frisby. PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985a.

Pollard, S. B., J. Porrill, J. E. W. Mayhew, and J. P. Frisby. Disparity gradient, Lipschitz continuity, and computing binocular correspondences. In *Proc. Third International Symposium on Robotics Research*, O. D. Faugeras and G. Giralt (Eds.), 19–26, Gouvieux France, 1985b. MIT Press.

Prazdny, K. Detection of binocular disparities. *Biological Cybernetics*, 52:387–395, 1985.

Sarachik, K. B. Visual navigation: Constructing and utilizing simple maps of an indoor environment. Technical Report AI-TR-1113, MIT, 1989.

Serey, B., and L. Matthies. Obstacle avoidance using 1-D stereo vision. unpublished CMU report, 1986.

Stewart, Jr., W. K. Multisensor modeling underwater with uncertain information. Technical Report AI-TR-1143, MIT, 1988.

Torre, V., A. Verri, and A. Fiumicelli. Stereo accuracy for robotics. In *Proc. Third International Symposium on Robotics Research*, O. D. Faugeras and G. Giralt (Eds.), 5–9, Gouvieux France, 1985. MIT Press.

Tsai, R. Y. An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 364–374, Miami Beach, FL, June 1986.

Tsai, R. Y., and R. K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand-eye calibration. In *Proc. Fourth International Symposium on Robotics Research*, 287–297, Santa Cruz, CA, 1987. MIT Press.

Tsai, R. Y., and R. K. Lenz. Real-time versatile robotics hand/eye calibration using 3D machine vision. In *Proc. IEEE International Conference on Robotics and Automation*, 554–561, Philadelphia, PA, April 1988.

Ullman, S. An approach to object recognition: Aligning pictorial descriptions. AI Memo 931, MIT, December 1986.

Viola, P. A. personal communication. MIT AI Lab, 1990.

Walpole, R. E., and R. H. Myers. *Probability and Statistics for Engineers and Scientists.* New York: Macmillan Publishing Company. 1978.

Wells, III, W. W. Visual estimation of 3-D line segments from motion—a mobile robot system. *IEEE Journal of Robotics and Automation*, 5(6), December 1989.

Witkin, A., D. Terzopoulos, and M. Kass. Signal matching through scale space. In *Proc. AAAI National Conference on Artificial Intelligence*, 714–719, Philadelphia, PA, August 1986.